Short Papers

of the

11th Conference on Cloud Computing, Big Data & Emerging Topics

HTTPS://JCC.INFO.UNLP.EDU.AR

@CONF_CC_BD_ET

JCC@LIDI.INFO.UNLP.EDU.AR

POSTGRADO
FACULTAD DE INFORMÁTICA

III-LIDI
INSTITUTO DE INVESTIGACIÓN EN INFORMÁTICA - LIDI

Short Papers of the 11th Conference on Cloud Computing Conference, Big Data & Emerging Topics (JCC-BD&ET 2023)

La Plata, Buenos Aires, Argentina.

June 27−29, 2023

Short Papers of the 11th Conference on Cloud Computing, Big Data & Emerging Topics / editado por Armando De Giusti, Marcelo Naiouf, Franco Chichizola, Enzo Rucci, Laura De Giusti - 1a ed. La Plata: Universidad Nacional de La Plata. Facultad de Informática, 2023.

Libro digital, PDF

ISBN 978-950-34-2271-7

9 789503 422717

Archivo Digital: descarga y online

ISBN 978-950-34-2271-7

1. Computación. 2. Actas de Congresos. 3. Conferencias.. I. De Giusti Armando, ed.

CDD 004.0711

Preface

Welcome to the short paper proceedings of the 11th Conference on Cloud Computing, Big Data & Emerging Topics (JCC-BD&ET 2023), held in a hybrid modality (both on-site and live online settings were allowed). JCC-BD&ET 2023 was organized by the III-LIDI and the Postgraduate Office, both from School of Computer Science of the National University of La Plata.

Since 2013, this event has been an annual meeting where ideas, projects, scientific results and applications in the cloud computing, big data and other related areas are exchanged and disseminated. The conference focuses on the topics that allow interaction between academia, industry, and other interested parties.

JCC-BD&ET 2023 covered the following topics: high-performance, edge and fog computing; internet of things; modelling and simulation; big and open data; machine and deep learning; smart cities; e-government; human-computer interaction; visualization; and special topics related to emerging technologies. In addition, special activities were also carried out, including 1 plenary lecture and 1 discussion panel.

In this edition, 11 short papers were accepted after the peer-review process. These short papers correspond to initial research with preliminary results, ongoing R+D projects, or postgraduate thesis proposals. The authors of these submissions came from the following 4 countries: Argentina, Chile, Colombia, and Spain. We hope readers will find these contributions useful and inspiring for their future research.

Special thanks to all the people who contributed to the conference's success: program and organizing committees, authors, reviewers, speakers, and all conference attendees.

June 2023

Armando De Giusti
Marcelo Naiouf
Franco Chichizola
Laura De Giusti
Enzo Rucci

Organization

General Chair

Marcelo Naiouf	Universidad Nacional de La Plata, Argentina.

Associate Chairs

Armando De Giusti	Universidad Nacional de La Plata and CONICET, Argentina.
Franco Chichizola	Universidad Nacional de La Plata, Argentina.
Laura De Giusti	Universidad Nacional de La Plata and CIC, Argentina.
Enzo Rucci	Universidad Nacional de La Plata and CIC, Argentina.

Program Committee

María José Abásolo	Universidad Nacional de La Plata and CIC, Argentina
José Aguilar	Universidad de Los Andes, Venezuela
Jorge Ardenghi	Universidad Nacional del Sur, Argentina
Javier Balladini	Universidad Nacional del Comahue, Argentina
Oscar Bria	Universidad Nacional de La Plata and INVAP, Argentina
Silvia Castro	Universidad Nacional del Sur, Argentina
Laura De Giusti	Universidad Nacional de La Plata and CIC, Argentina
Mónica Denham	Universidad Nacional de Río Negro and CONICET, Argentina
Javier Diaz	Universidad Nacional de La Plata, Argentina
Ramón Doallo	Universidade da Coruña, Spain
Marcelo Errecalde	Universidad Nacional de San Luis, Argentina
Elsa Estevez	Universidad Nacional del Sur and CONICET, Argentina
Pablo Ezzatti	Universidad de la República, Uruguay
Aurelio Fernandez Bariviera	Universitat Rovira i Virgili, Spain
Fernando Emmanuel Frati	Universidad Nacional de Chilecito, Argentina
Carlos Garcia Garino	Universidad Nacional de Cuyo, Argentina

Sponsors

III-LIDI

POSTGRADO
FACULTAD DE INFORMÁTICA

Facultad de INFORMÁTICA
UNIVERSIDAD NACIONAL DE LA PLATA

CIC
COMISIÓN DE
INVESTIGACIONES CIENTÍFICAS

CONICET

Ministerio de Ciencia,
Tecnología e Innovación Productiva
Presidencia de la Nación

Sistema Nacional
de Computación
de Alto
Desempeño

AGENCIA

RedUNCI

Table of Contents

Parallel and Distributed Computing

An approach to support generic topologies in distributed PSO algorithms in Spark

Xoán C. Pardo[0000−0001−8577−6980]1, Patricia González[0000−0003−0378−9222]1, Julio R. Banga[0000−0002−4245−0320]2, and Ramón Doallo[0000−0002−6011−3387]1

1 CITIC, Computer Architecture Group, Universidade da Coruña, Spain
{xoan.pardo,patricia.gonzalez,ramon.doallo}@udc.es
2 CSIC, Computational Biology Lab, Spain
j.r.banga@csic.es

Abstract. Particle Swarm Optimization (PSO) is a popular population-based search algorithm that has been applied to all kinds of complex optimization problems. Although the performance of the algorithm strongly depends on the social topology that determines the interaction between the particles during the search, current Metaheuristic Optimization Frameworks (MOFs) provide limited support for topologies. In this paper, we present an approach to support generic topologies in distributed PSO algorithms within a framework for the development and execution of population-based metaheuristics in Spark, which is currently under development.

1 Introduction

Parallel population-based metaheuristics [6] are common approaches to the optimization of large-scale problems because of their potential to obtain a satisfactory solution within a reasonable time. However, implementing scalable and efficient parallel metaheuristics requires knowledge of the underlying software platform and hardware architecture. Metaheuristic Optimization Frameworks (MOFs) provide customizable parallel implementations of the most popular metaheuristics, but there are significant differences in the level of support for parallelism and execution performance they provide [16, 23]. Furthermore, although distributed frameworks for Big Data, like Spark or Flink, have allowed to apply parallel metaheuristics at an unprecedented scale, MOFs support to these frameworks is very limited.

Particle Swarm Optimization (PSO) [14] is a population-based metaheuristic inspired by the collective behavior of flocks of birds that is very popular because it has a simple algorithmic description and has been successfully applied to a great variety of optimization problems. PSO represents the set of candidate solutions as a swarm of particles that traverse a multidimensional search space seeking the optimum. The interaction between particles during the search is determined by a social topology [15, 18] that dictates the grouping of particles into neighborhoods where information is shared. Even though topologies are fundamental in the performance of PSO, most MOFs only support the *gbest* topology.

In this paper, we present an approach that aims to overcome the aforementioned limitations by providing support for generic topologies in distributed PSO algorithms. The approach is implemented within a framework for the development and execution of population-based metaheuristics in Spark, which is currently under development. The rest of the paper is organized as follows: section 2 summarizes the related work; section 3 briefly introduces PSO and topologies; in section 4 the proposed approach is described; a preliminary experimental evaluation is presented in section 5; and section 6 concludes the paper.

2 Related work

To the best of our knowledge, there are only a few recent studies that compare MOFs and they only partially address the support for parallel and distributed computing. The comparative study in [16] shows that there are significant differences in the level of support, if any, that the frameworks provide for parallel and distributed computing. The analysis in [23] concludes that most MOFs only parallelize the evaluation of solutions, and only two provide support for both of the most common distributed models, i.e. the master-worker and the island-based models. Also, experimental evaluation found significant differences in performance under demanding configurations, both in terms of execution time and memory usage.

Furthermore, there are only a few proposals of MOFs that support Big Data frameworks. ECJ+Hadoop [4] is an enhancement to ECJ based on MapReduce that distributes the evaluation of solutions on Hadoop clusters using a *map* evaluator. jMetalSP [2] combines jMetal with different streaming engines such as Spark, Flink or Kafka. In Spark, RDDs are used to distribute the evaluation of solutions. HyperSpark [5] provides a configurable iterative workflow for the parallel execution in Spark of sequential metaheuristics as independent tasks, with or without cooperation between them. In [10] a framework to support the development of parallel evolutionary algorithms (PEAs) in Spark was proposed, and three PSO variants implemented in a unified way. With the exception of HyperSpark, all the proposals implement a master-worker model of parallelism, distributing only the evaluation of solutions.

With regard to PSO, some proposals implement parallel PSO variants on frameworks for Big Data. Examples using MapReduce are a constricted PSO, named MRPSO, in [19]; a cooperative PSO in [25]; or a quantum-behaved PSO in [27]. In [7] implementations in Hadoop and Spark are compared with a problem of energy optimization in buildings. Examples of implementations in Spark are a Cooperative co-Evolution PSO in [3]; a quantum-behaved PSO in [28]; or an hybridized island-based PSO in [12]. There are also applications to real problems, like the efficient utilization of water resources [17]; the training of Recurrent Neural Networks [26]; or clustering problems [1]. As far as we know, only MRPSO has support for different static or dynamic topologies, although only results for the *gbest* topology are reported in [19].

3 The PSO algorithm

The synchronous canonical PSO [13], in which all particles are updated at once, was used as a reference in this work, combined with different variants of the velocity update equation, i.e. the standard 2007 and the constricted and condensed constricted forms. Each particle stores its current position, velocity, fitness, and its best historical position and fitness. Positions and velocities are assumed to be D-dimensional vectors of real values.

3.1 Social topology

The influence of other particles is determined by the social topology, that dictates which particles are part of a neighborhood and the information shared between them. Originally, two topologies were proposed: (i) Global best *(gbest)*, all particles form a unique neighborhood; and (ii) Local best *(lbest)*, particles are influenced by its adjacent neighbors, usually two, following a *Ring* topology. In both of them, particles share their best position with their neighbors and the best of the neighborhood is used to update the velocity. As the topology highly influences the performance of the algorithm and there is no topology that is the best for all problems, different topologies have been studied [15, 18], including static (e.g. Von Newmann, Pyramidal, or Four Clusters), spacial (i.e. formed based on distance between particles), dynamic and adaptive structures, as well as other *models of influence* not based only on the best particle of the neighborhood, e.g. the *Full Informed PSO* (FIPS) [20].

We have analyzed the support for PSO topologies provided out of the box by four of the most widely used MOFs: Paradiseo (v3.0.0) [9], ECJ (v27) [24], HeuristicLab (v3.3.16) [11] and JMetal (v6.0) [8]. Our findings are summarized in the following:

- With the exception of JMetal, that does not support topologies out of the box, all the MOFs reviewed support at least the most common static topologies: Ring, Star and Random. ECJ and HeuristicLab also support dynamic topologies by randomly regenerating the topology at the end of each generation.
- Using the global and/or neighborhood best solution is the only model of influence supported.

4 An approach to support generic topologies

In this section we present a proposal to support generic topologies in distributed PSO algorithms that aims to overcome the limitations found in MOFs. An early version has been implemented within a framework for the development and execution of population-based metaheuristics in Spark, which is currently under development. In our approach, a social topology is composed by three elements:

1. A *self* boolean value, to indicate whether a particle should be included in its neighborhood.

Complete (*aka* GBest, Star, All), Directed Ring, Directed Scale Free, Generalized Petersen, G_{nm} Random, G_{np} Random Bipartite, Grid, HyperCube, KClusters, Kleinberg's Small World, Linear, Planted Partition, Pyramid, Random, Regular (with small-world shortcut [15]), Ring (*aka* LBest), Scale-Free Network, Square (*aka* VonNewman), Star (one particle in the center connected to all the others), Wheel, Windmill

Table 1: Topologies supported out of the box. With few exceptions, the topologies are parameterizable, e.g. for the Ring topology, the generalized version with k neighbors is provided.

2. A *topology shape*, represented as a DAG (*Directed Acyclic Graph*) of particle identifiers. Particles are assumed to have unique identifiers, and an edge $p_i \rightarrow p_j$ in the graph indicates that p_i is a neighbor of p_j and could share information with it. Table 1 shows the topology shapes currently supported. Custom topologies (listing 1.1) and importing topologies from GraphViz *.dot* files are also supported.
3. A *model of neighborhood influence*, represented as a function that maps particle identifiers to neighborhood influences. To support generic models of influence, two operations were defined by means of a Scala *trait* (listing 1.2): (i) *contribution*, a template method to collect the information shared by each particle with its neighborhood; and (ii) *neighborhoodInfluence*, a generic function builder, called on every swarm move, to instantiate the function that obtains the influence the neighborhood has on a particle. Different models are instantiated by passing different *collect* and *reduce* functions to *neighborhoodInfluence*. Currently, the models in [20] (i.e. Best, FIPS, wFIPS, wdFIPS, Self, and wSelf) are implemented. The instantiated function can then be used in any velocity update equation. For example, in the condensed form of the constricted equation, it would be called to get $\vec{I}_i^t$, the neighborhood influence on particle i at time t:

$$\vec{v}_i^{t+1} = \chi(\vec{v}_i^t + c_{max}(\vec{I}_i^t - \vec{x}_i^t)) \quad (1)$$

   where $\vec{x}_i^t$ and $\vec{v}_i^t$ are the position and velocity of the i-th particle at time t, χ is the constriction factor, and c_{max} is the upper limit of coefficients sum.

The proposal was designed with distributed PSO algorithms in mind. The *collect-reduce* approach shown in algorithm 1 was applied in the implementation of *neighborhoodInfluence*. It has two phases:

```
CustomRing {
  topology {
    self = true
    shape {
      name = Custom
      neighborhoods {
        0: [1, 2, 4],
        1: [2, 3, 0],
        2: [3, 4, 1],
        3: [4, 0, 2],
        4: [0, 1, 3]
      }
    }
    neighborhood_influence.name = wdFIPS
  }
}
```

Listing 1.1: Configuration of a custom *Ring* topology for a swarm of size N=5 with FIPS weighted by distance model of neighborhood influence and $Neighborhood(p_i) = \{p_i, p_{i+1\,mod\,N}, p_{i+2\,mod\,N}, p_{i-1\,mod\,N}\}$.

```
trait NeighborhoodOps {
  // method to collect the contribution of each particle to the neighborhood
  def contribution[T](pf:ParticleContributionFunction[T]):NeighborhoodContribution[T]
  // a generic factory method to instantiate the neighborhood influence function
  def neighborhoodInfluence[T,S](self:Boolean)(t:Topology)(
  collect:ParticleContributionFunction[T])(
  reduce:NeighborhoodContributionFunction[T,S]):NeighborhoodInfluenceFunction[S]
}
```

Listing 1.2: Operations added to a swarm to support generic models of neighborhood influence.

Algorithm 1 Pseudo-code of *neighborhoodInfluence.*

```
    Inputs: self; topology; collect(); reduce()
    Output: the neighborhood influence function
 1: $contrib$ = contribution(collect)          ▷ the contribution of each particle is collected
 2: for each $id$ in topology do
 3:     $neighbors$ = topology.predecessors($id$)          ▷ neighbors are the predecessors in the DAG
 4:     if self then
 5:         $neighborhood = id ++ neighbors$          ▷ the particle $id$ is included in the neighborhood
 6:     else
 7:         $neighborhood = neighbors$
 8:     end if
 9:     $influence[id]$ = reduce(contrib($id$), contrib($neighborhood$))   ▷ the influence the neighborhood has on the
    particle is calculated and stored in a map indexed by $id$
10: end for
11: return $id \Rightarrow influence[id]$          ▷ function that maps particle $ids$ to neighborhood influences
```

1. The *contribution* method (line 1) is called to collect the information contributed by each particle of the swarm. Different implementations are provided for the two swarm states, grouped or distributed, supported by our framework.
2. The *reduce* function (line 9) is then used to calculate the influence the neighborhood has on a particle from the contributions of the particle and its neighbors.

5 Experimental evaluation

Several experiments were performed to validate the proposal, as a preliminary evaluation focused on debugging and profiling the current implementation and verifying the genericness and correctness of the approach. All the experiments were carried out in a MacBook Pro M1 Pro with 10-core CPU and 16 GB RAM using the sequential version of the PSO algorithm implemented in our framework. Due to the space limitations, only a summary of the results is reported here. Configuration files, logs and detailed analysis of the results are available in a companion repository [22].

The first series of experiments performed reproduce those in [20]. The same methodology described in the paper was followed to evaluate the performance, iterations to criteria and proportion reaching criteria of six models of neighborhood influence (i.e. Best, FIPS, wFIPS, wdFIPS, Self, and wSelf) combined with five topologies (i.e. Square, Ring, 4-clusters, Pyramid and All), four of which are used in two different configurations, with and without including the target particle in the neighborhood. Only experiments with symmetrical initialization were reproduced. The results were obtained by combining the standarized individual results of five benchmark functions (i.e. Sphere, Rastrigin, Griewank -in two sizes-, Rosenbrock and Schaffer f6), that were executed 40 times each for each combination with the parameters provided in the paper.

Although all experiments run successfully, showing the genericness of the proposal, the results were quite different to those reported in [20] for all the dimensions evaluated. For example, the proportion of experiments reaching criteria, i.e. the proportion of runs that found the VTR (*Value To Reach*) within 10,000 iterations, is shown in Table 2. In general, the ratios of success are well below those reported in [20]. The best results were obtained by the wdFIPS model with a 99.6% ratio for the fully connected topologies, and the other FIPS variants with ratios higher than 91% for the UAll topology. The worst results were obtained by the Best model. The Self variants performed best in all cases, except for the fully connected topologies.

Since the values used for the PSO parameters are not given in [20] or whether any strategies were used to improve the performance of the algorithm, we decided to conduct a second series of experiments, reducing the search space of some functions and limiting the velocities and positions of the particles. Although this time the results showed more similarities, they were still worse in general.

To verify if the difference might be caused by differences in the PSO parameters, a second series of experiments were carried out to compare the proposal with the PSO implementation of ECJ using a two-sample Kolmogorov-Smirnov test. The experiments of the first series were repeated using the

	Square	Ring	4-Clusters	Pyramid	All	USquare	URing	UPyramid	UAll
Best	0.017	0.021	0.029	0.008	0.008	**0.038**	0.029	0.008	0.017
FIPS	0.167	0.167	0.217	0.283	0.663	0.167	0.167	0.167	**0.917**
wFIPS	0.167	0.167	0.167	0.167	0.625	0.167	0.167	0.167	**0.917**
wdFIPS	0.167	0.167	0.167	0.167	**0.996**	0.167	0.167	0.167	**0.996**
Self	0.417	0.546	0.479	0.475	0.375	0.571	**0.854**	0.446	0.383
wSelf	0.300	0.458	0.379	0.400	0.296	0.633	**0.804**	0.388	0.354

Table 2: Proportion of the first series of experiments reaching the VTR within 10,000 iterations. In bold are the best results for each model of neighborhood influence.

	Ring	Ring$_4$	All	Rand$_4$	URing	URing$_4$	UAll	URand$_4$
PSO-T	0.045	0.015	0.04	0.0	0.02	0.02	0.015	0.0
ECJ	0.04	0.05	0.03	0.005	0.01	0.015	0.005	0.0
Mendes	0.913		0.754		0.908		0.754	

Table 3: Ratio of success of our approach (PSO-T) and ECJ for the topologies evaluated. Results from [20] are also provided when available.

combinations supported by ECJ, i.e. the Best model of influence combined with the Ring, Random and All topologies, the first two with two different degrees, and the same benchmark functions except Schaffer f6. This time the results were very similar. The null hypothesis was rejected only in 6 out of the 160 combinations tested at significance level $\alpha = 0,043$ and 2 out of 160 at significance level $\alpha = 0,01$. This similarity can also be seen in Table 3, which shows a comparison of the ratio of success. As an anecdote, thanks to the experiments performed in this comparison, several bugs were detected in the PSO implementation of ECJ [21] and a pull request with the fix was submitted to the ECJ repository.

6 Conclusions

In this paper, an approach to support generic topologies in distributed PSO algorithms implemented within a framework for the development and execution of population-based metaheuristics in Spark has been presented. A preliminary experimental validation of the genericness and correctness of the approach was carried out using a sequential PSO implementation and different combinations of topologies and models of neighborhhood influence. To the best of our knowledge, no other MOF provides this level of genericness for PSO topologies. As future work, it is planned to perform an evaluation of the parallel performance of the approach and adding support for dynamic topologies in the near term.

Acknowledgments. This work was supported by the Ministry of Science and Innovation of Spain (PID2019-104184RB-I00, AEI, 10.13039/501100011033) and by Xunta de Galicia (Consolidation Program of Competitive Reference Groups, ref. ED431C 2021/30).

References

1. Al-Sawwa, J., Ludwig, S.A.: Parallel particle swarm optimization classification algorithm variant implemented with apache spark. Concurrency and Computation: Practice and Experience **32**(2), e5451 (2020). `https://doi.org/https://doi.org/10.1002/cpe.5451`
2. Barba-González, C., Nebro, A.J., Benítez-Hidalgo, A., García-Nieto, J., Aldana-Montes, J.F.: On the design of a framework integrating an optimization engine with streaming technologies. Future Generation Computer Systems **107**, 538–550 (2020). `https://doi.org/https://doi.org/10.1016/j.future.2020.02.020`
3. Cao, B., Li, W., Zhao, J., Yang, S., Kang, X., Ling, Y., Lv, Z.: Spark-based parallel cooperative co-evolution particle swarm optimization algorithm. In: 2016 IEEE International Conference on Web Services (ICWS). pp. 570–577 (2016). `https://doi.org/10.1109/ICWS.2016.79`
4. Chávez, F., de Vega, F.F., Lanza, D., Benavides, C., Villegas, J., Trujillo, L., Olague, G., Román, G.: Deploying massive runs of evolutionary algorithms with ecj and hadoop: Reducing interest points required for face recognition. The International Journal of High Performance Computing Applications **32**(5), 706–720 (2018). `https://doi.org/10.1177/1094342016678302`

5. Ciavotta, M., Krstić, S., Tamburri, D.A., Van Den Heuvel, W.J.: Hyperspark: A data-intensive programming environment for parallel metaheuristics. In: 2019 IEEE International Congress on Big Data (BigDataCongress). pp. 85–92 (2019). https://doi.org/10.1109/BigDataCongress.2019.00024
6. Crainic, T.: Parallel Metaheuristics and Cooperative Search, pp. 419–451. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-319-91086-4_13
7. Cui, L.: Parallel PSO in Spark. Master's thesis, Faculty of Science and Technology. University of Stavanger (2014)
8. Doblas, D., Nebro, A.J., López-Ibáñez, M., García-Nieto, J., Coello Coello, C.A.: Automatic design of multi-objective particle swarm optimizers. In: Dorigo, M., Hamann, H., López-Ibáñez, M., García-Nieto, J., Engelbrecht, A., Pinciroli, C., Strobel, V., Camacho-Villalón, C. (eds.) Swarm Intelligence. pp. 28–40. Springer International Publishing, Cham (2022)
9. Dreo, J., Liefooghe, A., Verel, S., Schoenauer, M., Merelo, J.J., Quemy, A., Bouvier, B., Gmys, J.: Paradiseo: From a modular framework for evolutionary computation to the automated design of metaheuristics: 22 years of paradiseo. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 1522–1530. Association for Computing Machinery, New York, NY, USA (2021), https://doi.org/10.1145/3449726.3463276
10. Duan, Q., Sun, L., Shi, Y.: Spark clustering computing platform based parallel particle swarm optimizers for computationally expensive global optimization. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) Parallel Problem Solving from Nature – PPSN XV. pp. 424–435. Springer International Publishing, Cham (2018)
11. Elyasaf, A., Sipper, M.: Software review: the heuristiclab framework. Genetic Programming and Evolvable Machines **15**(2), 215–218 (2014). https://doi.org/10.1007/s10710-014-9214-4
12. Fan, D., Lee, J.: A hybrid mechanism of particle swarm optimization and differential evolution algorithms based on spark. KSII Transactions on Internet and Information Systems **13**(12), 5972–5989 (2019)
13. Freitas, D., Lopes, L.G., Morgado-Dias, F.: Particle swarm optimisation: A historical review up to the current developments. Entropy **22**(3) (2020). https://doi.org/10.3390/e22030362
14. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks. vol. 4, pp. 1942–1948. IEEE (1995)
15. Liu, Q., Wei, W., Yuan, H., Zhan, Z.H., Li, Y.: Topology selection for particle swarm optimization. Information Sciences **363**, 154–173 (2016). https://doi.org/https://doi.org/10.1016/j.ins.2016.04.050
16. Lopes Silva, M.A., de Souza, S.R., Freitas Souza, M.J., de França Filho, M.F.: Hybrid metaheuristics and multi-agent systems for solving optimization problems: A review of frameworks and a comparative analysis. Applied Soft Computing **71**, 433–459 (2018). https://doi.org/https://doi.org/10.1016/j.asoc.2018.06.050
17. Ma, Y., an Zhong, P., Xu, B., Zhu, F., Lu, Q., Wang, H.: Spark-based parallel dynamic programming and particle swarm optimization via cloud computing for a large-scale reservoir system. Journal of Hydrology **598**, 126444 (2021). https://doi.org/https://doi.org/10.1016/j.jhydrol.2021.126444
18. McNabb, A., Gardner, M., Seppi, K.: An exploration of topologies and communication in large particle swarms. Faculty Publications. 869. (2009). https://doi.org/10.1109/CEC.2009.4983015
19. McNabb, A.W., Monson, C.K., Seppi, K.D.: Parallel pso using mapreduce. In: 2007 IEEE Congress on Evolutionary Computation. pp. 7–14 (2007). https://doi.org/10.1109/CEC.2007.4424448
20. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: simpler, maybe better. IEEE Transactions on Evolutionary Computation **8**(3), 204–210 (2004). https://doi.org/10.1109/TEVC.2004.826074
21. Pardo, X.C.: Ecj issue 82: bugs in the pso implementation, https://github.com/GMUEClab/ecj/issues/82
22. Pardo, X.C.: Experiments to validate the proposal of a generic topology for distributed PSO algorithms (Apr 2023). https://doi.org/10.5281/zenodo.7823049
23. Ramírez, A., Barbudo, R., Romero, J.R.: An experimental comparison of metaheuristic frameworks for multi-objective optimization. Expert Systems **n/a**(n/a), e12672 (2021). https://doi.org/https://doi.org/10.1111/exsy.12672
24. Scott, E.O., Luke, S.: Ecj at 20: Toward a general metaheuristics toolkit. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 1391–1398. GECCO '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319619.3326865
25. Wang, Y., Li, Y., Chen, Z., Xue, Y.: Cooperative particle swarm optimization using mapreduce. Soft Computing **21**(22), 6593–6603 (2017). https://doi.org/10.1007/s00500-016-2390-9
26. Wu, K., Zhu, Y., Li, Q., Han, G.: Algorithm and implementation of distributed esn using spark framework and parallel pso. Applied Sciences **7**(4) (2017). https://doi.org/10.3390/app7040353
27. Zhang, G., Li, Y., Chen, Z., Wang, Y., Jiao, L., Xue, Y.: A novel distributed quantum-behaved particle swarm optimization. Journal of Optimization **2017**, 4685923 (2017). https://doi.org/10.1155/2017/4685923
28. Zhang, Z., Wang, W., Gao, N., Zhao, Y.: Spark-based distributed quantum-behaved particle swarm optimization algorithm. In: Luo, Y. (ed.) Cooperative Design, Visualization, and Engineering. pp. 295–298. Springer International Publishing, Cham (2018)

Modelling Chronic Kidney Disease progression using ABM: a work in progress

Candelaria Alvarez[1], Jose Ibeas 2, Javier Balladini[3], and Remo Suppi[1]

[1] Universitat Autònoma de Barcelona, Dept. de Arquitectura de Computadores y Sistemas Operativos, 08193, Cerdanyola (Barcelona), España
{AnaCandelaria.Alvarez, Remo.Suppi}@uab.cat
[2] Clinical, Interventional and Computational Nephrology, I3PT, Parc Tauli University Hospital, Sabadell, Spain. jibeas@tauli.cat
[3] Universidad Nacional del Comahue, Neuquén, Argentina
javier.balladini@fi.uncoma.edu.ar

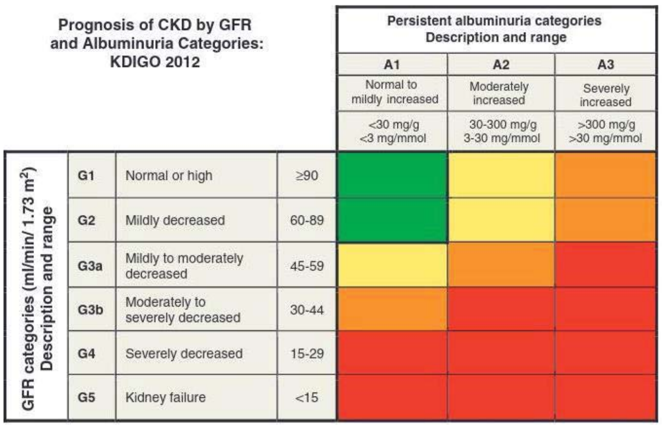
Abstract. Chronic kidney disease (CKD) is a major health problem worldwide. In Spain, the incidence of CKD has increased nearly 17% in ten years, from 2010 to 2020; and in 2020 it had a prevalence of 20% in adults over 60 years. This disease has some characteristics that make it complex; for instance, CKD has a late diagnosis, because the symptoms appear when it is already advanced, and they are not directly related to kidney problems. Furthermore, the causes of the disease are multiple; the most common are arterial hypertension and diabetes. In this context, we believe that simulation and agent-based modelling can provide new tools for assisting professionals in the decision-making process. Therefore, we are proposing the design of a model for CKD progression in order to predict the appearance of renal failure and its evolution in patients through the different states of the disease. Our work is divided into four main stages; and in this article, we present the tasks that have been carried out so far in the first stage which are related to data analysis to define the complexity of the problem.

Keywords: CKD · Agent-based modeling · High-performance simulation · Decision support system.

1 Introduction

Chronic kidney disease (CKD) is a public health problem worldwide. The latest report from the Spanish Society of Nephrology presented in March 2022 [12] indicates that in 2020 there were 141.4 new patients per million people in Spain requiring renal replacement therapy (RRT), representing a growth of almost 17% in the incidence of the disease over the last 10 years.

CKD is defined as kidney damage for a period of three or more months. The main indicators used are glomerular filtration rate (GFR) and albuminuria [16]. It is classified globally into 18 stages, as shown in Figure 1. CKD progression is unavoidable as patients suffer a continuous decline in their GFR. The rate of this decline is not linear, so sudden decreases cannot be anticipated.



Prognosis of CKD by GFR and Albuminuria Categories: KDIGO 2012				Persistent albuminuria categories Description and range		
				A1	A2	A3
				Normal to mildly increased	Moderately increased	Severely increased
				<30 mg/g <3 mg/mmol	30-300 mg/g 3-30 mg/mmol	>300 mg/g >30 mg/mmol
GFR categories (ml/min/ 1.73 m^2) Description and range	G1	Normal or high	≥90	🟩	🟨	🟧
	G2	Mildly decreased	60-89	🟩	🟨	🟧
	G3a	Mildly to moderately decreased	45-59	🟨	🟧	🟥
	G3b	Moderately to severely decreased	30-44	🟧	🟥	🟥
	G4	Severely decreased	15-29	🟥	🟥	🟥
	G5	Kidney failure	<15	🟥	🟥	🟥

Fig. 1. CKD categories by KDIGO guide [9]

There are no symptoms in the early stages of the disease, henceforth CKD tends to have a difficult diagnosis [3]. In consequence, under-diagnosis occurs in primary care patients; and as a consequence, kidney damage is normally high when the disease is diagnosed. By knowing the medical history and following-up with patients, it could be possible to try to predict the time of kidney damage appearance.

When CKD reaches its terminal stage (G5 GFR category), the patient loses the filtering capacity of his kidneys and must start having dialysis in order to remove uraemic toxins from the blood. Previously, a surgical intervention was necessary to perform the arterio-venous fistula which is needed for hemodialysis. In this context, it would be useful for doctors to know how quickly a patient will reach this stage, as it would allow a better planning of the surgical intervention.

A number of studies have been done related to CKD and informatics, and we classify them into two big groups. The first one [11, 13, 4] produces untraceable ML models for classifying an instance of a patient as either having a CKD diagnosis or not. They use datasets without historic information of each patient. In our research we propose the use of real historical data from Spanish patients and the definition of a traceable agent based model. The second group [14, 15, 5] uses the patient current state in order to predict risks probabilities in a period of years, such as death, need of renal replacement therapy (RRT) or cardiovascular diseases. Each study works with a subset of the CKD categories, generally with the most advanced stages. Our proposal includes all the stages from diagnosis to need of RRT.

2 Objectives and methodology

The main objective of this work is the development and validation of an agent-based model (ABM) for the prediction of CKD progression. The requirements are the following: simulate groups or types of patients with similar characteristics, use various data sources (such as personal history or test results) and validate the model with real data.

The methodology used will be incremental, dividing the work into four main stages: 1) Adequacy and filtering of real datasets on the evolution of CKD patients to classify them and define possible profiles, 2) Design of an ABM to predict disease progression, 3) Analysis, selection and/or definition of metrics and indices for evaluation and monitoring and 4) Model validation with real data and medical supervision.

3 Current work and future developments

The progress made on the first two stages can be divided into two parts:

1) Analysis of the problem complexity: We did an exploratory analysis in order to determine the magnitude and complexity of the problem. For this purpose, a dataset of patients diagnosed with CKD and COVID containing 9025 rows and 139 attributes was used. The analysis was carried out on the BigML platform [1], using its implementations of G-means algorithm [2] and then the Principal Component Analysis (PCA) [8].

G-means : It is a clustering algorithm presented in [6] that can be used when the correct number of clusters is unknown. Initially it runs the K-means algorithm with K=1. When it finishes, it uses the Anderson-Darling statistical test to determine whether the data assigned to each cluster has a Gaussian distribution. If it does not, it splits each cluster into 2 and reruns K-means with these new centroids. In case it does, the last split is discarded and the final result is the clusters of the previous iteration. G-means only has one parameter which is the significance level of the test statistic. It indicates how strict the test statistic should be when evaluating whether to accept or discard a cluster split. The lower the value, the stricter the test becomes and the more clusters it produces. In BigML, the parameter is called critical value, and can take values between 1 and 20. Furthermore, BigML's stopping criteria of the algorithm is slightly modified: it converges if there are multiple iterations that introduce new clusters but do not improve the quality of the clusters..

PCA: It is an unsupervised machine learning method which objective is finding the principal components that maximise the variance in the dataset. Then, it is possible to describe the data in a reduced number of dimensions by selecting a subset of these components. PCA is often used for dimensional reduction and data visualisation of large or complex datasets. In BigML it is possible to handle numeric and non-numeric data types, including text, categorical, items fields, as well as combinations of different data types.

By applying the G-means algorithm on the dataset we obtained 124 clusters, with critical value = 1, and 110 clusters, with critical value = 5. With these results we are able to observe that the problem is complex, since a large number of patient groups are detected with a clustering algorithm.

A PCA analysis was performed in order to evaluate the relevance of the variables within the dataset. A total of 139 components were obtained, where each component is a combination of all the attributes of the dataset. The 5 components with the highest percentages of variance obtained were the following: 10%, 7.7%, 3.2%, 2.82% and 2.7%. After adding the variance value of each component we obtain the percentage of the original data that can be represented. For example, with 20 components we obtain 50% of the original dataset. To reach almost the 70% we need 40 components, which represent 68.18% of the data. We can conclude that the problem is of large magnitude and also that we need to reduce te relevant variables for a patient.

2) Selection of an initial set of patient variables: After having sized the magnitude of the problem, we defined the initial patient variables and states of our model. The state diagram will have the 18 combinations of GFR and albuminuria indicated in Figure 1. One of the most important aspects in the definition of this diagram is the selection of the representative variables. For this selection we use the previous work of a research team member. In [7], the authors analysed a cohort of more than 10,000 patients over the age of 18 with 11 years of follow-up data. Using the XBGClassifier, they found that the 10 most relevant variables (from highest to lowest) were: cardiopathy, advanced CKD, vasculopathy, age, neoplasia, transplant, digestive pathology, estimated GFR and high blood pressure. In recent works on new datasets they have obtained a new list of variables: estimated GFR, age, microalbuminuria, BMI, HDL, glucose, urea, platelets, triglycerides and sodium. Because of this, we will have to analyze the variables using a dataset grouping both lists to determine which ones will be used for our first definition of the prediction model.

We also selected a simulation tool to implement the model. Scalability was taken into account, as we need to be able to model a large number of patients. Because of this, the Python library Mesa [10] was selected, which provides a framework for ABM. NetworkX will be used for the graphical representation, because it provides easy graph representation and manipulation of nodes and arcs.

Future developments: In order to accomplish all the stages mentioned, the future tasks are the following: 1) Expansion of the model, incorporating more input variables and defining the transitions between states. 2) Definition of metrics to evaluate the patient condition. 3) Validation of the model with real data and medical supervision.

4 Conclusions

This paper shows the work in progress of our research, which final objective is the design of a predictive model for the progression of CKD. The developed tasks are the analysis of the magnitude of the problem, using machine learning algorithms, and the selection of variables and states for the first version of the model. A preliminary and simplified model is currently being developed using Python & Mesa/NetworkX libraries.

Acknowledgements: This research has been supported by the Agencia Estatal de Investigacion (AEI), Spain and the Fondo Europeo de Desarrollo Regional (FEDER) UE, under contract PID2020-112496GB-I00 and partially funded by the Fundacion Escuelas Universitarias Gimbernat (EUG).

References

1. Bigml: Ml made beautifully simple for everyone, `https://bigml.com/`
2. Ashenfelter, A.: Divining the ‘k’ in k-means clustering (2015), `https://blog.bigml.com/2015/02/24/divining-the-k-in-k-means-clustering/`
3. Calderón-González, C., García-Fernández, N.: Enfermedad renal crónica: clasificación, etiopatogenia y factores pronósticos. Medicine - Programa de Formación Médica Continuada Acreditado **10**(79), 5364–5369 (2011). https://doi.org/10.1016/S0304-5412(11)70105-6
4. Dritsas, E., Trigka, M.: Machine learning techniques for chronic kidney disease risk prediction. Big Data and Cognitive Computing **6**(3) (2022). https://doi.org/10.3390/bdcc6030098
5. Grams, M.E., et al.: Predicting timing of clinical outcomes in patients with chronic kidney disease and severely decreased glomerular filtration rate. Kidney International **93**(6), 1442–1451 (2018). https://doi.org/10.1016/j.kint.2018.01.009
6. Hamerly, G., Elkan, C.: Learning the k in k-means. In: Proceedings of the 16th International Conference on Neural Information Processing Systems. p. 281–288. NIPS’03, MIT Press, Cambridge, MA, USA (2003)
7. Ibeas, J., Galles, O., Monill, et al.: Mo463: Machine learning-based prediction of mortality and risk factors in patients with chronic kidney disease developed with data from 10000 patients over 11 years. Nephrology Dialysis Transplantation **37**(Supplement_3) (05 2022). https://doi.org/10.1093/ndt/gfac070.077
8. Jesus, M.: Principal component analysis (pca): Dimensionality reduction! (2018), `https://blog.bigml.com/2018/12/05/principal-component-analysis-pca-dimensionality-reduction/`
9. KDIGO: Clinical practice guideline for the evaluation and management of chronic kidney disease (2012), `https://kdigo.org/guidelines/ckd-evaluation-and-management/`
10. Masad, D., Kazil, J.L.: Mesa: An agent-based modeling framework. In: SciPy (2015)
11. Mondol, C., et al.: Early prediction of chronic kidney disease: A comprehensive performance analysis of deep learning models. Algorithms **15**(9) (2022). https://doi.org/10.3390/a15090308
12. de Nefrología, S.E.: La enfermedad renal crónica en españa (2022)
13. Prakash, S., et al.: Ensemble nonlinear support vector machine approach for predicting chronic kidney diseases. Computer Systems Science and Engineering **42**(3), 1273–1287 (2022). https://doi.org/10.32604/csse.2022.021784
14. Tangri, N., et al.: A predictive model for progression of chronic kidney disease to kidney failure. JAMA **305**(15), 1553 (Apr 2011). https://doi.org/10.1001/jama.2011.451
15. Tangri, N., et al.: Multinational assessment of accuracy of equations for predicting risk of kidney failure: A meta-analysis. JAMA **315**(2), 164 (Jan 2016). https://doi.org/10.1001/jama.2015.18202
16. Webster, A., Nagler, E., Morton, R., Masson, P.: Chronic kidney disease. The Lancet **389** (11 2016). https://doi.org/10.1016/S0140-6736(16)32064-5

Brief Performance Portability Analysis of a Matrix Multiplication Kernel on Multiple Vendor GPUs

Manuel Costanzo[1], Enzo Rucci[1]*, Carlos García-Sánchez[2], and Marcelo Naiouf[1]

[1] III-LIDI, Facultad de Informática, UNLP – CIC.
La Plata (1900), Bs As, Argentina
{mcostanzo,erucci,mnaiouf}@lidi.info.unlp.edu.ar
[2] Dpto. Arquitectura de Computadores y Automática, Universidad Complutense de Madrid. Madrid (28040), España
garsanca@dacya.ucm.es

Abstract. The heterogeneous computing paradigm has led to the need for portable and efficient programming solutions that can leverage the capabilities of various hardware devices, such as NVIDIA, Intel, and AMD GPUs. This study evaluates the performance and portability of the SYCL and CUDA languages for a matrix multiplication (MM) application across different GPU architectures. The experimental work showed that, while the CUDA implementation outperforms the SYCL implementation on NVIDIA devices due to optimizations provided by the `nvcc` compiler, the latter implementation demonstrated remarkable code portability to other GPU architectures, such as AMD and Intel. Furthermore, the architectural efficiency percentages obtained on AMD and Intel GPUs showed consistency with the results observed on NVIDIA devices.

Keywords: oneAPI · SYCL · GPU · CUDA· Performance portability

1 Introduction

In the last decade, the quest to improve the energy efficiency of computing systems has fueled the trend toward heterogeneous computing and massively parallel architectures [1]. Nowadays, GPUs can be considered the dominant accelerator, and Nvidia, Intel, and AMD are the biggest manufacturers. In the 4th quarter of 2022, Intel and AMD had 9% of the market, with Nvidia dominating the discrete graphics card market at 82%. By including integrated and embedded graphics, Intel had 71% of the market, Nvidia 17% and AMD 12% [3].

Focusing on the programming aspect, CUDA is the most popular GPU programming language. However, CUDA codes only run on Nvidia GPUs. This fact

* Corresponding author.
[3] https://www.pcgamer.com/intel-is-already-matching-amd-for-gaming-graphics-market-share/

imposes severe limitations to code portability, also affecting maintenance, extension, and development cost. One effort to face this issue is SYCL [4], a new open standard from Khronos Group. SYCL is a royalty-free, cross-platform abstraction layer that allows the programmer to write single-source C++ host code including accelerated code expressed as functors. In this sense, the same SYCL code can run on various hardware platforms, including CPUs, GPUs, and FPGAs. In this way, SYCL seeks to reduce development and maintenance costs and also improve programming productivity.

In this context, while reaching functional portability is already hard, performance portability becomes a major challenge. In this paper, we evaluate the functional and performance portability of two GPU-accelerated implementations of a matrix multiplication (MM) kernel across Intel, Nvidia, and AMD GPUs using Marowka's method [2].

2 Background

2.1 SYCL and the oneAPI Programming Ecosystem

SYCL is a cross-platform programming model based on C++ language for heterogeneous computing and features asynchronous task graphs, hierarchical parallelism, buffers defining location-independent storage, automatic overlapping kernels and communications, and interoperability with OpenCL, among other characteristics [3]. Recently, Intel announced the *oneAPI* programming ecosystem that provides a unified programming model for a wide range of hardware architectures. At the core of the oneAPI environment is the Data-Parallel C++ (DPC++) programming language, which can be summarized as C++ with SYCL. Additionally, DPC++ also features some vendor-provided extensions that might be integrated into these standards in the future [4]. Last, oneAPI provides different programming utilities, including a compatibility tool (`SYCLomatic`) that facilitates the migration to the SYCL-based DPC++ programming language.

2.2 Performance portability

According to Penycook [5], *performance portability* refers to *"A measurement of an application's performance efficiency for a given problem that can be executed correctly on all platforms in a given set"*. These authors define two different performance efficiency metrics: *architectural efficiency* and *application efficiency*. The former represents the ability of an application to utilize hardware efficiently and is a fraction of "peak" theoretical hardware performance; while the latter represents the ability of an application to use the most appropriate implementation and algorithm for each platform, and is a fraction of the best-observed performance.

The metric for performance portability presented by Penycook [5] was later reformulated by Marowka [2] to address some of its flaws. The latter is presented

[4] `https://www.khronos.org/registry/SYCL/specs/sycl-2020/pdf/sycl-2020.pdf`

next. Formally, for a given set of platforms H from the same architecture class, the performance portability $\bar{\Phi}$ of a case-study application α solving problem p is:

$$\bar{\Phi}(\alpha, p, H) = \begin{cases} \frac{\sum_{i \in H} e_i(\alpha,p)}{|H|} & \text{if } i \text{ is supported } \forall i \in H \\ \text{not applicable (NA)} & \text{otherwise} \end{cases}$$

where $e_i(\alpha, p)$ is the performance efficiency of case-study application α solving problem p on the platform i.

3 Experimental Work and Results

3.1 Case-Study Applications: Matrix Multiplication (MM)

Two GPU-accelerated implementations of matrix multiplication (MM) kernel were considered for the performance portability evaluation:

- `CUDA`: this version was extracted from the CUDA Demo Suite [5]. This app computes a MM using shared memory through a tiled approach and loop unrolling technique to increase throughput.
- `SYCL`: this code is based on the implementation provided in [6], which represents an SYCL-equivalent, migrated version of the previous one.

It is important to note that, according to Nvidia, the code "*It has been written for clarity of exposition to illustrate various CUDA programming principles, not with the goal of providing the most performant generic kernel for matrix multiplication.*" [6]

3.2 Experimental Results

The experiments were performed on eight systems equipped with different GPUs. The main features of these systems are described in Table 1. A single workload was configured for MM ($nIter = 10$; $wA, wB, hA, hB = \{16384\}$). Finally, to run SYCL code on Nvidia and AMD GPUs, several modifications had to be made to the build, as it is not supported by default [7]. After these modifications, it was possible to run DPC++ code on an Nvidia GPU, but using the Clang++ compiler (`nvcc 11.7`, `clang 16.0`).

Table 2 presents the GFLOP/s and architectural efficiencies of both CUDA and SYCL codes on the experimental platforms. On the one hand, it becomes evident that CUDA outperforms SYCL using all Nvidia GPUs. In particular, CUDA version runs (on average) 1.2× faster than SYCL. This superior performance can be primarily attributed to the fact that `nvcc`, performs a more efficient

[5] `https://docs.Nvidia.com/cuda/cuda-samples/`
[6] `https://docs.Nvidia.com/cuda/cuda-c-programming-guide/index.html`
[7] https://intel.github.io/llvm-docs/GetStartedGuide.html

Table 1: Experimental platforms

CPU		**GPU**		
Processor	**RAM Memory**	**Vendor**	**Model (Architecture)**	**GFLOPS peak (SP)**
Intel Xeon E5-2695	16 GB		GTX 980 (Maxwell)	4980
Intel Xeon E5-2695	16 GB		GTX 1080 (Pascal)	8872
Intel Core i5-7400	64 GB	Nvidia	RTX 2070 (Turing)	7464
Intel Core i5-10400F	64 GB		RTX 3070 (Ampere)	20313
Intel Xeon Gold 6138	64 GB		Tesla V100 (Volta)	14131
Intel Core i9-9900K	32 GB	Intel	Arc 770 (Gen 12.5)	19660
Intel Xeon iE5-1620	64 GB	AMD	RX 6700 XT (RDNA 2.0)	13214

Table 2: GFLOP/s and architectural efficiencies of both CUDA and SYCL codes on the experimental platforms.

Platform		**CUDA**		**SYCL**	
GPU	**GFLOP/s peak**	**GFLOP/s achieved**	**Arch. eff.**	**GFLOP/s achieved**	**Arch. eff.**
GTX 980	4980	552	11.1%	430	8.6%
GTX 1080	8872	603	6.8%	556	6.3%
RTX 2070	7464	1011	13.6%	810	10.9%
RTX 3070	20313	1316	6.5%	1084	5.3%
Tesla V100	14131	1582	11.2%	1345	9.5%
Arc 770	19660	×	NA	1836	9.3%
RX 6700 XT	13214	×	NA	1553	11.8%

code translation than `clang++` when it comes to shared memory accesses [8], causing SYCL code to use more registers and perform additional computation. On the other hand, architectural efficiencies are low for both code versions (8% on average). This behavior is related to the educative aspect of the original code that was detailed in Section 3.1.

Performance portability of both CUDA and SYCL codes is presented in Table 3. First, it becomes evident that SYCL code provides higher functional portability, successfully running on different hardware vendor platforms. Moreover, CUDA fails to demonstrate the same level of adaptability, just being able to run on Nvidia GPUs. Second, both codes present a similar performance efficiency when executing on the different supported GPUs, demonstrating their performance portability.

[8] `https://support.codeplay.com/t/poor-performance-on-matrix-multiplication/575/2?u=mcostanzo`

Table 3: Performance portability of both CUDA and SYCL codes on the experimental platforms.

	$\Phi(\alpha, p, H)$	
Platform set (*H*)	**CUDA**	**SYCL**
Nvidia	9.8%	8.1%
Intel	NA	9.3%
AMD	NA	11.8%
Nvidia ∪ AMD	NA	8.7%
Nvidia ∪ Intel	NA	8.3%
Intel ∪ AMD	NA	10.5%
Nvidia ∪ AMD ∪ Intel	NA	8.8%

3.3 Discussion

While SYCL code proved to be slower than its CUDA counterpart in this study, it showcased performance portability across a wider range of GPU vendors, highlighting its versatility and potential. However, it is important to note that the observed performance difference between SYCL and CUDA codes does not occur in all cases; [7, 8, 9] show that SYCL codes can achieve the same or even better performance than CUDA versions.

4 Conclusions and Future Work

In this paper, we have evaluated both the performance and portability of SYCL and CUDA languages for a MM application on Nvidia, Intel, and AMD GPUs. The main findings of this study can be summarized as follows:

- The performance comparison between the SYCL and CUDA implementations on Nvidia devices revealed that the latter outperforms the former due to the optimizations applied by the `nvcc` compiler.
- We have successfully demonstrated the code portability of the SYCL implementation to other GPU architectures, such as AMD and Intel. Moreover, the architectural efficiency percentages obtained on these GPUs were found to be consistent with those observed on Nvidia devices.

In summary, this brief study highlights the potential of SYCL as a performance-portable alternative to CUDA for the development of high-performance computing applications. Although the current performance of SYCL on Nvidia GPUs may be lower than that of CUDA, this gap will decrease as SYCL compilers continue to improve.

Future work focuses on exploring the use of SYCL in different application domains. This could provide valuable insights into its performance and portability features in a broader context, enabling a more comprehensive understanding of its strengths and limitations.

References

[1] H. Giefers et al. "Analyzing the energy-efficiency of sparse matrix multiplication on heterogeneous systems: A comparative study of GPU, Xeon Phi and FPGA". In: *2016 IEEE ISPASS*. 2016, pp. 46–56.

[2] Ami Marowka. "Reformulation of the performance portability metric". In: *Software: Practice and Experience* 52.1 (2022), pp. 154–171. DOI: `https://doi.org/10.1002/spe.3002`.

[3] Ronan Keryell and Lin-Ya Yu. "Early Experiments Using SYCL Single-Source Modern C++ on Xilinx FPGA". In: *Proceedings of the IWOCL '18*. Oxford, UK: ACM, 2018. DOI: `10.1145/3204919.3204937`.

[4] S. Christgau and T. Steinke. "Porting a Legacy CUDA Stencil Code to oneAPI". In: *2020 IEEE IPDPSW*. May 2020, pp. 359–367. DOI: `10.1109/IPDPSW50202.2020.00070`.

[5] S.J. Pennycook, J.D. Sewall, and V.W. Lee. "Implications of a metric for performance portability". In: *Future Generation Computer Systems* 92 (2019), pp. 947–958. ISSN: 0167-739X. DOI: `https://doi.org/10.1016/j.future.2017.08.007`.

[6] Manuel Costanzo et al. "Early Experiences Migrating CUDA codes to oneAPI". In: *IX Jornadas de Cloud Computing, Big Data & Emerging Topics*. 2021.

[7] Manuel Costanzo et al. "Migrating CUDA to oneAPI: A Smith-Waterman Case Study". In: *Bioinformatics and Biomedical Engineering*. Cham: Springer International Publishing, 2022, pp. 103–116. ISBN: 978-3-031-07802-6. DOI: `10.1007/978-3-031-07802-6_9`.

[8] Zheming Jin and Jeffrey S. Vetter. "Performance Portability Study of Epistasis Detection Using SYCL on NVIDIA GPU". In: *Proceedings of the 13th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. BCB '22. Northbrook, Illinois: ACM, 2022. ISBN: 9781450393867. DOI: `10.1145/3535508.3545591`.

[9] Goutham Kalikrishna Reddy Kuncham, Rahul Vaidya, and Mahesh Barve. "Performance Study of GPU applications using SYCL and CUDA on Tesla V100 GPU". In: *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. 2021, pp. 1–7. DOI: `10.1109/HPEC49654.2021.9622813`.

Parallel Model of Online Sequential Extreme Learning Machines for Classification Problems with Large-Scale Databases

Elkin Gelvez-Almeida[1,3], Ricardo J. Barrientos[2](✉), Karina Vilches-Ponce[2], and Marco Mora[2]

[1] Doctorado en Modelamiento Matemático Aplicado, Universidad Católica del Maule, Talca, Chile

[2] Laboratory of Technological Research in Pattern Recognition (LITRP), Facultad de Ciencias de la Ingeniería, Universidad Católica del Maule, Chile
{rbarrientos,mmora,kvilches}@ucm.cl

[3] Facultad de Ciencias Básicas y Biomédicas, Universidad Simón Bolívar, San José de Cúcuta, Colombia
elkin.gelvez@unisimon.edu.co

Abstract. Nowadays, the sizes of databases in real-world applications are around TeraByte or PetaByte. Therefore, training neural networks in reasonable times is challenging and requires high-cost computational architectures. OS-ELM is a variant of ELM, proposed for real-world applications. This algorithm allows training with new data using the previous results without reusing the previous dataset. In this work, we present a parallel model of OS-ELM for classification problems using large-scale databases. The model consists of training several OS-ELM using multi-threaded programming. The training dataset is distributed according to the number of working threads. Then, the test dataset is classified by all pre-trained OS-ELMs. Finally, the test dataset is classified using a frequency criterion. Preliminary results show that increasing the number of threads decreases the training time without significantly affecting the test accuracy of each OS-ELM.

Keywords: parallel computing · high performance computing · extreme learning machine · fingerprint classification.

1 Introduction

Extreme learning machine (ELM) is an artificial neural network (ANN) algorithm proposed for single hidden layer feedforward neural networks (SLFN). The algorithm consists of assigning random weights and biases to the hidden layer and analytically calculating the weights of the output layer using the Moore–Penrose generalized inverse matrix [3]. ELM has increased its popularity and acceptance in the scientific community due to the simplicity of the model and its generalization capacity in classification and regression problems. However, the increase in database size has been a challenge for these networks, since training times increase significantly and high-cost computational architectures are required.

Online sequential extreme learning machine (OS-ELM) is a variant of ELM proposed to address real-world problems where training data are obtained chunk-by-chunk or one-by-one [6]. This model is trained with new samples using the previous results. This methodology allows train with large-scale databases but with a high training time. OS-ELM variants that decrease the training time without affecting the accuracy were proposed in the literature. Lan et al. [5] proposed an ensemble of online sequential extreme learning machine (EOS-ELM). This algorithm constructs P OS-ELMs to form EOS-ELM, where each OS-ELM trains with new data at each incremental step. Zhai et al. [9] proposed EOS-ELM for large data set classification. Mirza et al. [7] proposed EOS-ELM for class imbalance and concept drift. Finally, Huang et al. [4] proposed a parallel EOS-ELM based on MapReduce. These previous works have inspired our proposal.

2 Online Sequential Extreme Learning Machine

In this section, we present a brief description of the preliminaries for this work starting with ELM. Let $\aleph$ be an arbitrary training set $\aleph = \{(\boldsymbol{x}_i, \boldsymbol{t}_i) | \boldsymbol{x}_i \in \mathbb{R}^n, \boldsymbol{t}_i \in \mathbb{R}^m\}$ with $i = 1, \ldots, N$, an activation function $g : \mathbb{R}^n \to \mathbb{R}^m$, and L neurons in the hidden layer with $L < N$ [3], SLFN training algorithm is given by:

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i g(\boldsymbol{w}_i \cdot \boldsymbol{x}_j + b_i) = \boldsymbol{t}_j, \qquad j = 1, \ldots, N, \tag{1}$$

$\boldsymbol{w}_i$ and b_i are the i-th weights and biases of the hidden layer, respectively, $\boldsymbol{\beta}_i$ is the i-th weight of the output layer, and $\boldsymbol{w}_i \cdot \boldsymbol{x}_j$ represents the inner product of $\boldsymbol{w}_i$ and $\boldsymbol{x}_j$ [3]. Equation (3) can be written in matrix form as $\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T}$, where $\boldsymbol{H}$ matrix is the output matrix of the hidden layer of the neural network [3]. The $\boldsymbol{\beta}_i$ weights of the output layer are calculated analytically using $\boldsymbol{\beta} = \boldsymbol{H}^{\dagger}\boldsymbol{T}$ [1], where $\boldsymbol{H}^{\dagger}$ is the Moore-Penrose generalized inverse matrix of $\boldsymbol{H}$.

We now give a brief description of OS-ELM. This algorithm is a variant of ELM for real-world applications [6]. OS-ELM has two phases: the initialization phase and the sequential learning phase. The *initial phase* uses the initial chunk of training data $\aleph_0$ with N_0 number of samples. Then, the initial weights $\boldsymbol{\beta}^0$ of the output layer are computed using the following equation:

$$\begin{cases} \boldsymbol{P}_0 = (\boldsymbol{H}_0^T \boldsymbol{H}_0 + \boldsymbol{I}/C)^{-1} \\ \boldsymbol{\beta}^0 = \boldsymbol{P}_0 \boldsymbol{H}_0^T \boldsymbol{T}_0 \end{cases}, \tag{2}$$

where H_0 is the initial output matrix of the hidden layer and C is the regularization parameter.

The *sequential learning phase* introduced the $(k+1)$-th chunks of training data. This phase makes sequential learning where $\boldsymbol{\beta}^{k+1}$ is calculated using the following equation:

$$\begin{cases} \boldsymbol{P}_{k+1} = \boldsymbol{P}_k - \boldsymbol{P}_k \boldsymbol{H}_{k+1}^T (\boldsymbol{I} + \boldsymbol{H}_{k+1} \boldsymbol{P}_k \boldsymbol{H}_{k+1}^T)^{-1} \boldsymbol{H}_{k+1} \boldsymbol{P}_k \\ \boldsymbol{\beta}^{k+1} = \boldsymbol{\beta}^k \boldsymbol{P}_{k+1} \boldsymbol{H}_{k+1}^T (\boldsymbol{T}_{k+1} - \boldsymbol{H}_{k+1} \boldsymbol{\beta}^k) \end{cases}, \tag{3}$$

The OS-ELM training is completed when the chunks of training data are finished.

3 Materials and method

In this section, we present the materials and methods used in this work. All previous experiments were carried out in a server with 2×Intel(R) Xeon(R) Gold 6238R CPUs @ 2.20GHz and 128GB of RAM, using C++ programming language and the OpenMP library for parallel programs with shared memory. This server is part of the computer cluster of the Laboratory of Technological Research in Pattern Recognition (LITRP) of the Universidad Católica del Maule, Talca, Chile.

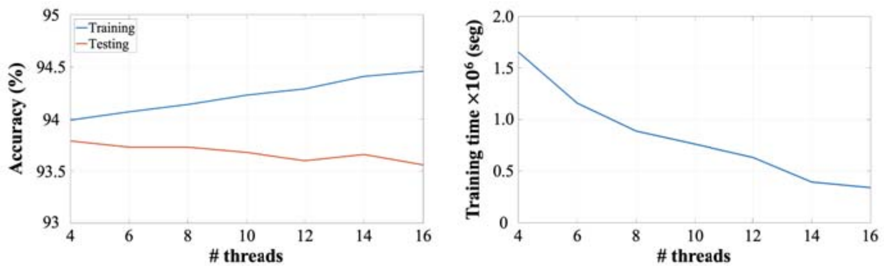
About the dataset, we used a synthetic fingerprint dataset. We employed a feature extractor to transform the fingerprints into feature descriptors. As a result of the feature extractor, vectors with 202 numerical values representing the fingerprints were generated. In addition, each descriptor (vector) has a label corresponding to the class (five fingerprint types) [8]. We balance the database for the previous experiments. Thus, the frequency of occurrence of the classes is the same (20% for class). On the other hand, we considered the neurons in the hidden layer and the regularization parameter C as hyperparameters. For the estimation, we used a data set with 60,000 samples. Experimental results show that the best accuracy is obtained with 2,000 to 5,000 neurons in the hidden layer and $C = 10$ [2]. The proposed model is presented below.

3.1 Parallel model of online sequential extreme learning machines

First, the model is related to the parallel training of a set of OS-ELM. This model consists of independent OS-ELMs that train on several threads in parallel. Then, the accuracy of each OS-ELM is averaged to compare the results using different amounts of threads. In this model, we have used a dataset with 1,000,000 samples. We used 70% of the samples for training (700,000) and 30% for testing (300,000). We divide the data set according to the number of OS-ELMs. We use a multi-thread algorithm with a one-to-one relation between threads and cores to avoid resource conflicts by the operating system [2]. Finally, we use a selection criterion based on the results of each OS-ELM. We selected the highest frequency as the selection criterion. A test dataset is introduced by each trained OS-ELM, then the highest frequency label is selected from the OS-ELMs estimations.

4 Preliminary results

In this section, we present the preliminary results of the proposed model. We used from 4 to 16 threads. The dataset was distributed according to the quantity of threads used in the experiments. The results (time and accuracy) are averaged using a simple mean to present an overview of the proposed model performance. Fig. 1a shows the training and test accuracy as the number of threads increases. The size of the datasets for each thread decreases as the threads increase since the dataset is distributed according to the threads quantity. We can see that the training accuracy increases as the threads quantity does, but the test accuracy decreases. However, the test accuracy goes from 93.79% with 4 threads to 93.56% with 16, which does not reflect a significant change.



(a) (b)

Fig. 1: Preliminary results using 5,000 neurons in the hidden layer and the regularity parameter C=10. (a) Training and testing accuracy with different numbers of threads, and (b) training time in seconds with different numbers of threads.

Regarding the training time, Fig. 1b shows the results as the threads quantity increases. Training time decreases as the threads increase. The decrease in training time is due to the threads increase, as each OS-ELM is trained with a smaller dataset size. Regarding the accuracy and training time results, we see that it significantly decreases the training time without affecting the accuracy. Finally, the accuracy is expected to increase as the threads quantity increase using the frequency criterion.

5 Conclusions

This paper proposed a parallel model of online sequential extreme learning machines for classification problems with large-scale databases. This model distributed the dataset in various OS-ELMs trained in parallel. Then, the testing dataset is classified using a frequency criterion according to the results of each OS-ELM. The preliminary results show that the training time decreases when the threads number increases. Testing accuracy also is not affected when the threads quantity increases. The preliminary results suggest that it is appropiate to increase the threads quantity to decrease the training time, since it does not affect the accuracy of the test. However, it requires a computational architecture with a suitable core quantity. Finally, we expected that the accuracy to increase when applying the frequency criterion.

5.1 Future work

In future work, we plan to increase the threads number to identify the moment that decreases the accuracy. In addition, we will analyze the results using this unbalanced data set, simulating a population in the real world. We will also use other databases and ELM variants to test our model in different applications. In addition, we will propose a parallel mathematical model to compute the Moore-Penrose generalized inverse using multiple threads. This model aims to decrease the training time of ELM and their variants since it is the slowest procedure. Finally, computing the Moore-Penrose generalized inverse with a parallel model will allow us to design an OS-ELM model based on a parallel processing strategy as a pipeline.

Acknowledgement

This work was funded by the National Agency for Research and Development (ANID) / Scholarship Program / NATIONAL DOCTORAL SCHOLARSHIPS / 2020 - 21201000. The authors of the paper also thank Research Project FONDECYT REGULAR 2020 Nº 1200810 "Very Large Fingerprint Classification Based on a Fast and Distributed Extreme Learning Machine," National Agency for Research and Development (ANID), Government of Chile.

References

1. Gelvez-Almeida, E., Baldera-Moreno, Y., Huérfano, Y., Vera, M., Mora, M., Barrientos, R.: Parallel methods for linear systems solution in extreme learning machines: an overview. In: Journal of Physics: Conference Series. vol. 1702, p. 012017. IOP Publishing (2020)
2. Gelvez-Almeida, E., Barrientos, R.J., Vilches-Ponce, K., Mora, M.: Parallel training of a set of online sequential extreme learning machines. In: 2022 41st International Conference of the Chilean Computer Science Society (SCCC). pp. 1–4. IEEE, Santiago (2022)
3. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: A new learning scheme of feedforward neural networks. In: 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541). vol. 2, pp. 985–990. IEEE, Budapest (2004)
4. Huang, S., Wang, B., Qiu, J., Yao, J., Wang, G., Yu, G.: Parallel ensemble of online sequential extreme learning machine based on MapReduce. Neurocomputing **174**, 352–367 (2016)
5. Lan, Y., Soh, Y.C., Huang, G.B.: Ensemble of online sequential extreme learning machine. Neurocomputing **72**, 3391–3395 (2009)
6. Liang, N.Y., Huang, G.B., Saratchandran, P., Sundararajan, N.: A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Transactions on Neural Networks **17**, 1411–1423 (2006)
7. Mirza, B., Lin, Z., Liu, N.: Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift. Neurocomputing **149**, 316–329 (2015)
8. Zabala-Blanco, D., Mora, M., Barrientos, R.J., Hernández-García, R., Naranjo-Torres, J.: Fingerprint classification through standard and weighted extreme learning machines. Applied Sciences **10**, 4125 (2020)
9. Zhai, J., Wang, J., Wang, X.: Ensemble online sequential extreme learning machine for large data set classification. In: 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 2250–2255. IEEE, San Diego (2014)

A Preliminary study of vehicular traffic distribution in Great Mendoza area. Use of IoT tools.

Osvaldo Marianetti[1,4], Pablo Godoy[1,2,3], Ernesto Chediak1[1], Daniel Fontana [1,4] and Carlos García Garino[1,2]

Universidad Nacional de Cuyo, Mendoza, Argentina.
[1] Facultad de Ingeniería, [2] ITIC, [3] Facultad de Ciencias Exactas y Naturales
[4] Universidad de Mendoza. Facultad de Ingeniería, Mendoza, Argentina

osvaldo.marianetti@ingenieria.uncuyo.edu.ar, pablodgodoy@ingenieria.uncuyo.edu.ar, ernestochediack@gmail.com, cgarcia@itu.uncu.edu.ar

Abstract. This paper discusses the use of Radar Sensor Network at Great Mendoza area in order to register and to store vehicular traffic information. Both the network features and the software used in order to analyze and post process the captured information are provided. The main issues of vehicular traffic flow at Great Mendoza area are described from a qualitative point of view.
Preliminary results registered with a Radar Sensor Networks placed at Godoy Guaymallén department are discussed as well. Based on the obtained results it can be expected that this kind of tools be very useful in order to alleviate vehicular traffic bottlenecks and related problems.

Keywords: Vehicular Traffic, Radar Sensor Network, IoT, Great Mendoza Area.

1 Introduction.

The appearance of Internet of Things (IoT) [1] allow us to analyze different problems of practical interest. For instance, the vehicular traffic flow in large cities, a problem whose analysis and modeling is typical of the field of Smart Cities.

This paper presents and discusses the implementation of the radar sensor network, placed at Guaymallén department located in Great Mendoza zone.

This work is framed within the context of the research project "Studies on the contribution of technologies such as Blockchain, NFT and VPN to security in IOT platforms" [2,3].

The work is organized as follows: First, the qualitative description of some vehicular traffic problems in the area of Great Mendoza is provided. The radar sensor network installed at Guaymallén municipality is described in section 2. Some preliminary results obtained are discussed in section 3. Finally, the conclusions of the work and the corresponding bibliography are presented.

2 Great Mendoza area vehicular traffic issues

The Province of Mendoza, located in the west of the country, see figure 1, has around 2 million inhabitants. Approximately half of them inhabit the called Great Mendoza, see figure 1, which includes the departments of Capital, Guaymallén, Las Heras, Godoy Cruz, Maipú and Luján de Cuyo. The Capital (Mendoza City) and Godoy Cruz departments are the smallest in area and have the highest density housing. It is important to note that GreatMendoza is the fourth urban nucleus in Argentina, after AMBA, Great Córdoba and Great Rosario.

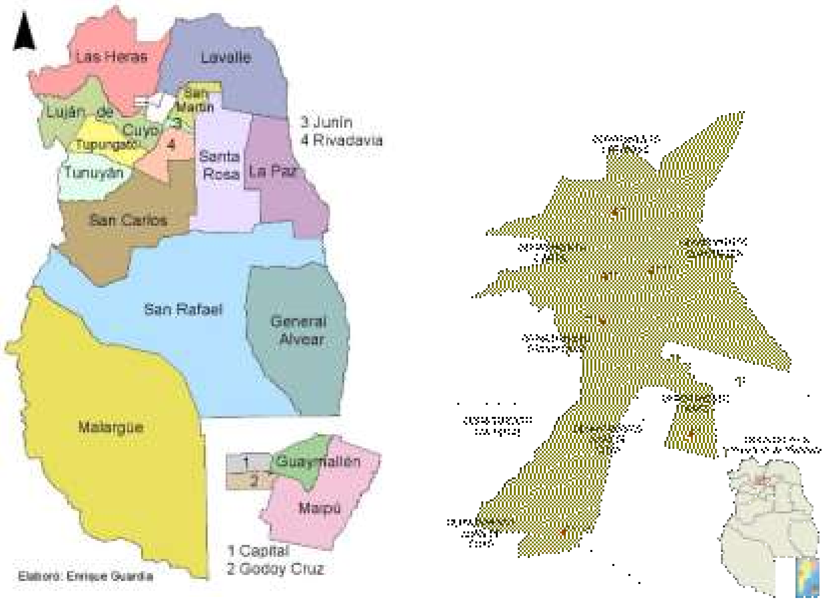


Fig. 1. Mendoza province and Great Mendoza area maps.

For different reasons: entry of employees to the public administration, opening of shops, beginning of the school day and in universities, among others, in the area of Great Mendoza, morning and afternoon traffic peaks occur. The incoming traffic flows to the Capital department, takes place between 7 A.M and 9 A.M., approximately. The flow of outgoing traffic starts at around 5 P.M.

3 Radar sensor network.

In this section the technical characteristics of Guaymallén Radar Sensor Network is provided. The Godoy Cruz infrastructure is quite similar to the Guaymallén one.

The different radars nodes are connected using a proper network in order in order to download the registered information as well as to do maintenance tasks.

In this case, for each radar node available in the network a 4G link connect them to a server that stores the information. Previously as can be seen in Figure 2, a 3G link was used. This link is set up by means of a Huawei model E173 USB modem equipped with a SIM card with an active line and available mobile data.

A TP-Link router model TL-MR3020 version 3.20 manages the connection for each radar sensor node. The original factory router operating system has been replaced by OpenWrt version 22.03.2 [4]. This alternartive operating system allows the connection as a client to an OpenVPN network as well as Internet link through the aforementioned 4G modem, in addition to other useful features for this application.

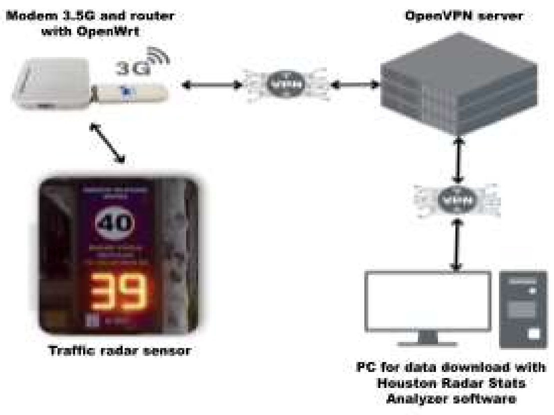


Fig. 2. Radar Sensor Network scheme.

The sensor module is a Doppler effect radar, brand Houston RADAR, Model SS400, whose operating frequency is 24.125 GHz +/- 50MHz. The antenna power is 5mW or higher and the polarization is linear. The operating temperature is in the range between -20ºC and 85ºC. It has a measurement range: 2.1Km/H – 161Km/h, which can be configured and a resolution of +/- 0.1 Km/h, with an accuracy:+/- 0.5% of the read value. The maximum detection distance covers a range from 90m (small vehicles) to 150m (large vehicles) traveling at speeds between between 20Km/h and 88Km/h.

The connection to the OpenVPN server will allow the radar to have a known fixed IP with which it can be accessed to download the desired information using the Houston Radar Stats Analyzer Software [5].

4 Preliminary results.

This section presents the results of the traffic measurement carried out between 07/20/22 and 08/04/22, on Avellaneda street in the Department of Guaymallén. During the registered period, 46,782 vehicles circulated, with a daily average of about 3,000 vehicles. In particular, 2,902 vehicles per day if 5 days are considered and 3,016 vehicles per day if 7 days are taken into account.

The graph of Figure 3 shows the hourly distribution of the vehicles for the 07/25/22 week. In 01/08/2022. The x-axis represents the considered time in hours and the y-axis accounts for the number of registered vehicles. This information is presented for each day in between Monday 7/25/2022 until Sunday 31/7/2022.

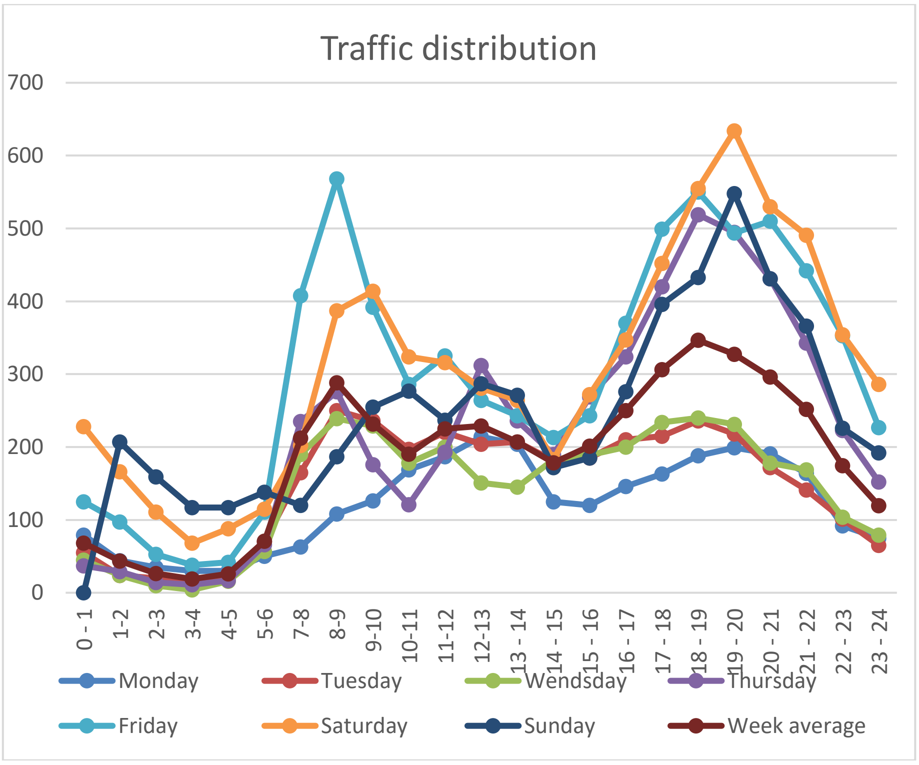


Fig. 3. Traffic hourly distribution for incoming vehicles.

In the graph of Figure 3 two different peaks can be clearly seen. One at early hours, in between 7 am and 9 am, and the other at the evening and night, from 5 pm until 9 pm. It is important to point out that the vehicular traffic increase at night during Friday and Saturday. Even at Saturday evening a larger amount of vehicles in comparison with working days can be observed. There is also a valley with a very low vehicular flow in the early morning hours. On the other hand, a marked decrease in registered traffic between 11 a.m. and 3 pm it is shown in the graphic. In this sense, it is important to say that business and school operation hours at Great Mendoza is not continuous.

In the graph of Figure 5 the week average 85% speed hourly distribution is shown. The 85% metric considers only the 85% of the total registers. The 15% not considered values belongs to the extreme upper and lower speed values. As can be observed in the graph the average speed values are larger at early hours. The limit speed is 45 km/h that is keep mainly at working hours.

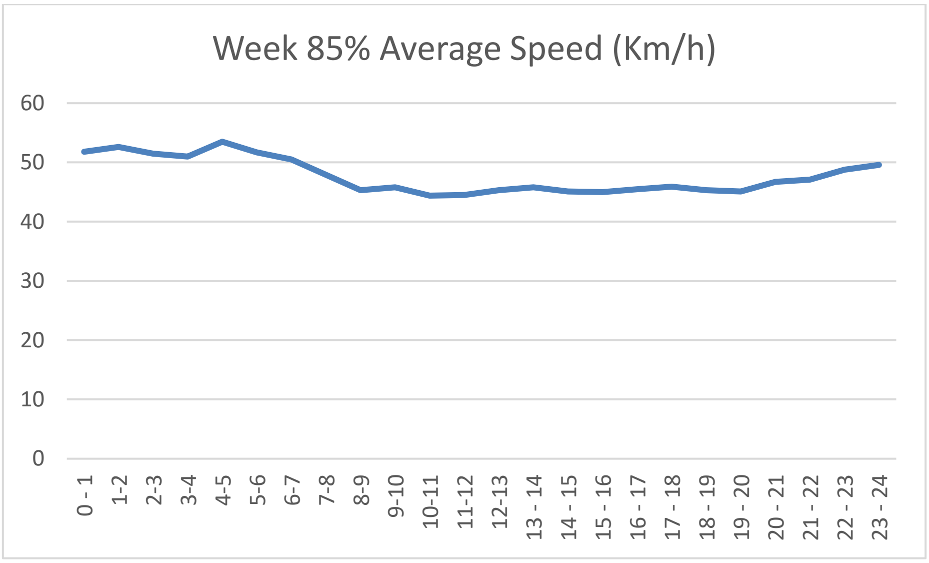


Fig. 5. Week 85% Average speed hourly distribution

5. Conclusion.

A radar sensor network and its implementation has been discussed in this paper. Their hardware, communications and software characteristics has been provided as well. Both functional prototypes are currently functional in two different departments at Great Mendoza area. The informed results belongs to a radar network available at Guaymallen department.

The preliminary results obtained are useful in order to model and estimate vehicular traffic flow. In this sense, the implemented radar sensor networks can be considered as a promising proof of concept.

Among other possibilities, Big Data analysis [6] of registered traffic can be a very useful tool in order to model and predict vehicular traffic flow.

6. Acknowledgments.

The assistance and cooperation provided by Guaymallen and Godoy Cruz City Hall departments is acknowledged. This work has partially granted by ITIC and SIIP UN-Cuyo Project "Estudios del Aporte de tecnologías como Blockchain, NFT y VPN a la seguridad en plataformas de IOT".

References

1. B. Mazon-Olivo, A. Pan." Internet of Things: State-of-the-art, Computing Paradigms and Reference Architectures". IEEE Latin America Transactions, VOL. 20, N°.1, January 2022.
2. Marianetti: Proyecto Estudios del Aporte de tecnologías como Blockchain, NFT y VPN a la seguridad en plataformas de IOT. Director O. Marianetti. SIIP, UNCUYO, 2022-2024.
3. O. Marianetti, P. Godoy, E. Chediak, D. Fontana. Estudios del Aporte de tecnologías como Blockchain, NFT y VPN a la seguridad en plataformas de IOT. In press. Proceedings of XV WICC 2023. UNNOBA, Junin, April 13-14th, 2023.
4. OpenWrt Proyect: WRT 22.03.04. https://openwrt.org. Accesed on April 12th, 2023.
5. Houston-Radar: Houston Radar Analyzer Software. https://houston-radar.com/products/software/stats-analyzer-configuration-software/. Accesed on April 12th, 2023.
6. Elgendy, N., Elragal, A. (2014). Big Data Analytics: A Literature Review Paper. In: Perner, P. (eds) Advances in Data Mining. Applications and Theoretical Aspects. ICDM 2014. Lecture Notes in Computer Science(), vol 8557. Springer, Cham. https://doi.org/10.1007/978-3-319-08976-8_16O.

Data analysis in telecommunication services

Marcelo Dante Caiafa[1], Ariel Aurelio[1], Alejandro Bevilacqua[1]

[1] Universidad Nacional de La Matanza, Florencio Varela 1903, Buenos Aires, Argentina
{macaiafa,aaurelio,abevilacqua}@unlam.edu.ar

Abstract. The data has become a main source of knowledge for organizations. Its adequate treatment allows to obtain valuable information. The professional link between university and industry is the goal of our academic work. It is intended to register an articulation project based on data analysis. It is an initiative based on a collaboration between a company and the academy. The aim is using the knowledge and skills of the students to improve the efficiency of the company by automating a process and at the same time bring computer engineering students closer to the productive environment.
The work is based on Python and its libraries for data processing of a telecommunication services. The project is based on an exploratory, correlation and segmentation study of data processing for telecommunication services. The goal is to build a customer profile with high dropout potential.
The work´s intention is to add value to decision makers in a data-based organization and verify the contribution that new tools can offer to students and teachers in the ICT sector (information and communication technologies).

Keywords: Data analysis, ICT, telecommunications, churn.

1 Introduction

The digital transformation is generating in the last decades big changes in the way of producing and consuming goods and services [1]. In this context, the information management allows organizations to improve their processes, "data is the new currency that sustains fundamental changes in the fourth industrial revolution" [2].

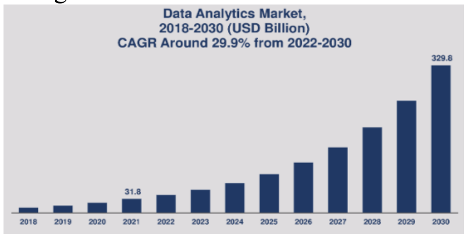


Fig. 1. Data Analytics Market (Acumen R&C 2021)

A data-oriented organizational culture has advantages in decision-making because it is based on evidence [3][16]. Some studies [4] indicate organizations with a data-driven approach improve their productivity and profitability. According to international consultants [5], the size of the global data analytics market represented USD 31.8 billion by 2021 and they estimate it will reach USD 329.8 billion in 2030. It would register a compound annual growth rate (CAGR) of 29.9% in that period.

Different types of analytics are classified in:

a) Descriptive analytics: It illustrates outcome data collected over a time interval.

b) Diagnostic analytics: it looks for the root cause of a problem.

c) Predictive analytics: it uses past data to make forecasts.

d) Prescriptive analytics: it involves machine learning techniques to forecasting.

Although there are different analytical maturity models, the following chart lists different types of analysis. These can be considered as correlative stages of the same project, which as the level of complexity progresses increases the contribution value [6].

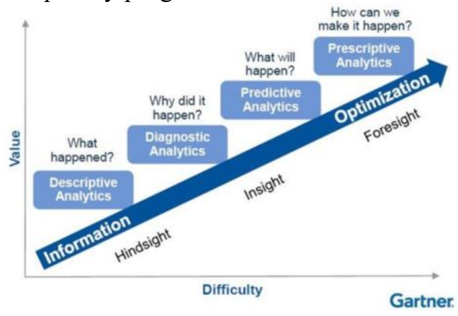


Fig. 2. Analytical Madurity Model (Gartner 2012)

The study is based on the articulation of the academic-professional world. It focuses on a project of exploratory analysis of data from telecommunications services. In the planning of the project, the recommendations of different authors [7], [8] are taken as a reference. Although a cited author [9] advances in detail for the R language, the present work was carried out in Python´s libraries. To facilitate the project´s development, the data analysis was grouped into three main stages [10].

Data Collection Stage: the automated data capture eliminates the manual efforts required for data entry. It was implemented by scraping process on CRM internal web site using Beautifulsoup.

Data Treatment Stage: In this instance the Pandas and Numpy libraries were used for data cleaning, identification of variables, construction of a data dictionary, analysis of relationships between variables, statistical description of data.

Data Presentation Stage: The information was visualized by Matplotlib and Seaborn libraries. They were so useful for summary of observations and results presentation.

A typical characteristic of EDA (Exploratory Data Analysis) is that it focuses more on the process than on the theory. A handful of entry-level analysis techniques (descriptive statistics, correlations, and basic visualizations) can be learned [11]. This kind of projects is not only about the implementation of technological infrastructure but due to their potential impact, they are strategic to support decision-making based on evidence.

The research question that guides the work is how to develop data analysis project that links engineer students to the telecommunications service sector. In this framework, the objectives are:

1. To document the process to identify a typical customer profile with the highest dropout rate, based on the most representative variables detected.

2. To build methodological proposal for EDA project with forecasting capabilities.

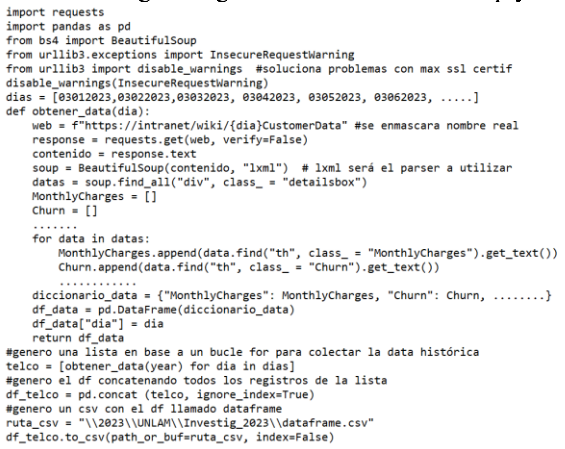
2 Work Development

This research project is structured in five instances executed sequentially. In the first instance, the tasks of selecting the software tool were carried out. In the second instance, the context and requirements are validated. The data collection is processed by scraping and the indicators are defined. The third instance is about data processing. They are cleaned, ordered and the statistical analysis is executed. In the stage number four, the

operations related to presentation of results are executed. The correlation matrix is built. The relevant variables are identified and validated with heat map. The graphs for presentation reports are elaborated. The last instance is the elaboration of conclusions and new proposals for future work.
Software selection had initially many options. R, ELK, powerBI were analyzed. Python 3.9.12 release as a programming language was defined as being the most appropriate due to its features: high-level language, dynamically typed, multiplatform, open source, object-oriented language, its big community and extensive libraries, among others [12].

2.1 Data Collection

In this instance, the context of the problem to be solved and its impact were studied. In other words, they inquired about the value that the sought solution would provide. This meant studying the relationship between customer lifecycle and churn rate. The costs related to retaining an existing client vs. the costs of attracting a new one and the different loyalty and cross-selling strategies were also studied deeply.



```
import requests
import pandas as pd
from bs4 import BeautifulSoup
from urllib3.exceptions import InsecureRequestWarning
from urllib3 import disable_warnings  #soluciona problemas con max ssl certif
disable_warnings(InsecureRequestWarning)
dias = [03012023,03022023,03032023, 03042023, 03052023, 03062023, .....]
def obtener_data(dia):
    web = f"https://intranet/wiki/{dia}CustomerData" #se enmascara nombre real
    response = requests.get(web, verify=False)
    contenido = response.text
    soup = BeautifulSoup(contenido, "lxml")  # lxml será el parser a utilizar
    datas = soup.find_all("div", class_ = "detailsbox")
    MonthlyCharges = []
    Churn = []
    .......
    for data in datas:
        MonthlyCharges.append(data.find("th", class_ = "MonthlyCharges").get_text())
        Churn.append(data.find("th", class_ = "Churn").get_text())
        ............
    diccionario_data = {"MonthlyCharges": MonthlyCharges, "Churn": Churn, ........}
    df_data = pd.DataFrame(diccionario_data)
    df_data["dia"] = dia
    return df_data
#genero una lista en base a un bucle for para colectar la data histórica
telco = [obtener_data(year) for dia in dias]
#genero el df concatenando todos los registros de la lista
df_telco = pd.concat (telco, ignore_index=True)
#genero un csv con el df llamado dataframe
ruta_csv = "\\2023\\UNLAM\\Investig_2023\\dataframe.csv"
df_telco.to_csv(path_or_buf=ruta_csv, index=False)
```

Fig. 3. Scrapping Process

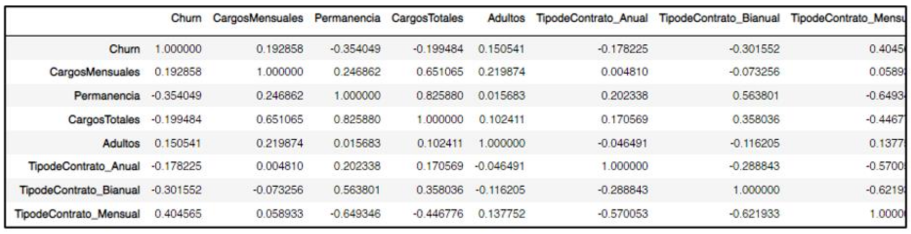
Data were collected by web scraping process on the intranet web page which is fed daily by CRM server (customer relationship manager) [13].

2.2 Data Proccesing

Various manipulations of the data set are performed here. First, the data type is consolidated for each variable. Missing data and outliers are detected, ordered and consolidated for subsequent statistical analysis, and categorical variables are mapped to numerical variables to facilitate their processing. Null and inconsistent data are removed. These tasks are executed with Pandas and Numpy libraries, according to good practices [14]. The variables are ordered, some of them are renamed and others are redefined based on the indicators. This case study is based in historical data with churn rate 26%.

2.3 Data Exploration

Statistical analysis is carried out to determine the most representative variables, correlation matrix is built and most relevant variables for churn prediction are identified.



	Churn	CargosMensuales	Permanencia	CargosTotales	Adultos	TipodeContrato_Anual	TipodeContrato_Bianual	TipodeContrato_Mensu
Churn	1.000000	0.192858	-0.354049	-0.199484	0.150541	-0.178225	-0.301552	0.4045
CargosMensuales	0.192858	1.000000	0.246862	0.651065	0.219874	0.004810	-0.073256	0.0589
Permanencia	-0.354049	0.246862	1.000000	0.825880	0.015683	0.202338	0.563801	-0.6493
CargosTotales	-0.199484	0.651065	0.825880	1.000000	0.102411	0.170569	0.358036	-0.4467
Adultos	0.150541	0.219874	0.015683	0.102411	1.000000	-0.046491	-0.116205	0.1377
TipodeContrato_Anual	-0.178225	0.004810	0.202338	0.170569	-0.046491	1.000000	-0.288843	-0.5700
TipodeContrato_Bianual	-0.301552	-0.073256	0.563801	0.358036	-0.116205	-0.288843	1.000000	-0.6219
TipodeContrato_Mensual	0.404565	0.058933	-0.649346	-0.446776	0.137752	-0.570053	-0.621933	1.0000

Fig. 4. Correlation Matrix

Finally, the most representative variables for the dropout rate are validated by heat map which is shown.

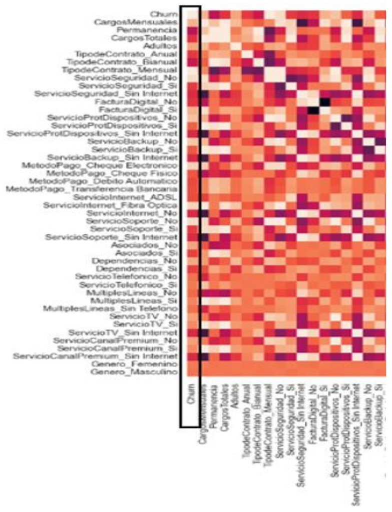


Fig. 5. Partial view of Heat map

3 Results

It is observed the clients with highest unsubscribe requests are those with highest monthly charges and lower than average tenure.

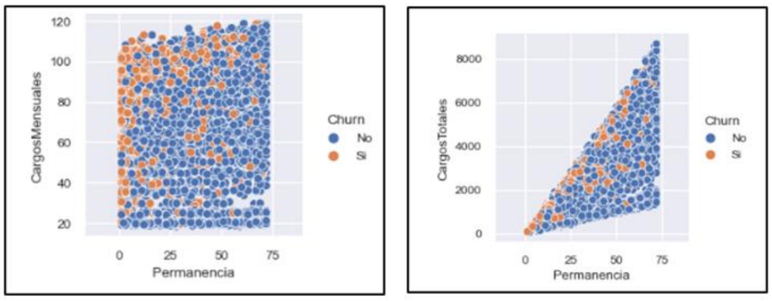


Fig.6. Interrelación de variables (cargos mensuales y totales vs permanencia)

4 Conclusions

The most representative variables in the telecommunications service customer abandonment rate were typified: type of contract (monthly), technical support (not contracted), payment method (electronic check) and online security (not contracted). It was possible to detect and ponder the most relevant causes that give early signs of a possible change contractual conditions. This allows preventive, constant and continuous analysis to avoid loss of assets such as the client portfolio. It gave scalability and dynamics to descriptive data analysis, defines it a first step in the data science study. For future work, it is proposed to continue with the development of the analytical maturity model, based on this descriptive analysis, advancing to the predictive one with machine learning, through SciPy libraries. Additionally, this model could be applied in other careers, other than engineering, that use data sets linked to their specific disciplines, to integrate particular knowledge and skills from their productive environments.

Competing interests

The authors have declared that no competing interests exist.

Authors' contribution

"MC conceived the idea and conducted the experiments; MC, AA and AB analyzed the results and revised the manuscript. All authors read and approved the final manuscript."

5 References

[1] S Schwab. La cuarta revolución industrial. World Economic Forum. Ed Debate, 2016.
[2] R. Privdeville. Prepare for data Revolution. Data-driven world. Armanino, 2019.
[3] M. Schwartz. War & Peace & IT: Business Leadership, Technology and Success in the digital age. Portland. Ed IT Revolution Press, 2019.
[4] A. McAfee, & Brynjolfsson. The Management Revolution. Harvard Business Review. 2012.
[5] Acumen, Research & Consulting. Global Data Analytic Market. 2022. Disponible en https://www.acumenresearchandconsulting.com/data-analytics-market
[6] W. Jensen. Statistics=Analytics?. 2021. Quality Engineering, Inc., Flagstaff, Arizona, p 7
[7] C, O'Neil & R. Schutt. Doing Data Science: Straight Talk from the Frontline. O'Reilly Media Inc, California (2013)
[8] J. Saltz & Shamshurin, I.: Exploring the process of doing data science via an ethnographic study of a media advertising company. In: 2015 IEEE International Conference on Big Data pp. 2098–2105. IEEE (2015)
[9] Wickham, H., Grolemund, G.: R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. O'Reilly Media Inc, California (2016)
[10] S. Van Daele & G. Jansseswillen. Identifying the Steps in an Exploratory Data Analysis. ICPM(international Conference on process Mining (2022) p 526. Ed. Springer
[11] M. Courtney. Exploratory Data Analysis in Schools: A Logic Model to Guide Implementation. (2021) IJEPL (International Journal Education Policy & Leadership. Volume17(4)
[12] W. Bel, Algoritmos y estructuras de datos en Python. Facultad C y Tec. 2020. p17. Ed Uader
[13] S. Mukhiya & U. Ahmed. Hands-On Exploratory Data Analysis with Python: Perform EDA techniques to understand your data. Ed. Packt 2020.
[14] W. McKinney. Python for Data Analysis: Data Wrangling with Pandas & Numpy. Third Edition. Ed. O´Reilly. 2022
[15] A. Pajankar. Hands-on Matplotlib. Learn Plotting and Visualizations with Python 3. Ed Apress 2021.

Artificial Intelligence

Comparing and evaluating tools for Sentiment Analysis

Franco M. Borrelli[1,2][0000-0002-1185-4461] and Cecilia Challiol[1,3][0000-0001-5140-0264]

[1] LIFIA, Facultad de Informática, UNLP, La Plata, Buenos Aires, Argentina
[2] UNLP Master's Scholarship, Argentina
[3] CONICET, Argentina
{fborrelli,ceciliac}@lifia.info.unlp.edu.ar

Abstract. Sentiment analysis is a process of identifying and extracting personal information from textual data. It has become essential for businesses and organizations to understand customers' opinions, emotions, and attitudes toward their products, services, or brands. While creating a custom sentiment analysis model can provide tailored results for specific datasets, it can also be time-consuming, resource-intensive, and require a high level of expertise in machine learning. Some tools offer a faster and more accessible alternative to users without a background in machine learning to create a custom model. However, researchers and practitioners usually do not know how to choose the best tool for each domain. This paper compares and evaluates some sentiment analysis tools' differences, considering how they were built and how suitable they are for analyzing sentiments on some specific topics. In particular, this paper focuses on four popular sentiment analysis tools for Python: TextBlob, Vader, Flair, and HuggingFace Transformers.

Keywords: Sentiment Analysis, TextBlob, Vader, Flair, HuggingFace Transformers, Ruled-based approach, Machine Learning.

1 Introduction

Sentiment analysis [1] is a natural language processing (NLP) technique that automatically identifies and extracts personal information from a text. This technique is used to analyze large amounts of text data, such as customer reviews, social media posts, and news articles, to determine the overall sentiment or attitude expressed in the text. Sentiment analysis has become essential for businesses, researchers, and individuals to understand how people feel about a particular topic or product [2].

There are two options for performing sentiment analysis: creating a custom model or using pre-built tools. The first can provide more tailored results for specific datasets; it can also be time-consuming, resource-intensive, and require high expertise (as knowledge in machine learning). On the other hand, some pre-trained tools that have been tested for accuracy and reliability can be used out of the box. They require minimal coding experience, making them accessible to a broader range of users who may not have a background in machine learning. However, there still needs to be documented research comparing and evaluating their effectiveness in specific areas of interest without going into detail on how they work. In addition, the few existing studies often focus on comparing up to two of these tools [3].

This paper aims to reduce the mentioned gap by providing a comparative evaluation of popular sentiment analysis tools for Python. To do that, this paper compares and evaluates the performance of four tools on a standardized dataset and provides insights into their strengths and weaknesses to help practitioners or researchers decide which tool might be more suitable for their problem. Two evaluations on standardized datasets from different areas of interest are comparatively evaluated for these tools.

This paper is structured as follows. Section 2 describes a brief survey of the tools under analysis. Section 3 presents the results of two evaluations to determine the effectiveness of each tool. Conclusions are mentioned in Section 4.

2 Background

Sentiment analysis tools [1] have unique strengths and limitations, and understanding these differences is critical when selecting the suitable tool for a particular task. There are some critical aspects to pay attention to that condition this selection.

The first issue is the methodology used for the prediction. There are two main approaches [4]: rule-based and machine learning. The first relies on a dictionary of words labeled as: positive, negative, or neutral. A sentence is tokenized, and each token is compared with the available words (in the dictionary). Then, a combination function such as sum or average is used to make the final prediction. This method only focuses on individual words, ignoring the context in which they are used [3], so, for example, sarcasm is often misunderstood. In contrast, machine learning methods use algorithms[1] to learn from data and identify text patterns that indicate sentiment. These methods typically require large amounts of labeled data for training but can provide higher accuracy and more flexibility. Predictions are usually slower because the computational algorithm is usually much more complex. An analysis comparing the performance of both approaches is presented in [4]. Hybrid models (that combine aspects of these two approaches) can further improve the results [5].

The second critical aspect is associated with data from different domains used in each tool (such as vocabulary present in product reviews or movie reviews) to determine what is positive and negative. These differences can lead to varying levels of effectiveness depending on the specific topic being analyzed. The last issue is that some tools give a general weighting (between -1 and 1), whereas others indicate whether the result is positive or negative and the degree of confidence (from 0 to 1).

This paper focuses on four sentiment analysis tools for Python: TextBlob[2] (using its two alternatives), Vader[3], Flair[4], and HuggingFace Transformers[5]. Table 1 presents vital distinctions among each of these tools.

1 For example, Naïve Bayes, Support Vector Machines, or Neural Networks.
2 TextBlob, https://textblob.readthedocs.io/en/dev/, last access: 14/03/2023.
3 Vader, https://github.com/cjhutto/vaderSentiment, last access: 14/03/2023.
4 Flair, https://github.com/flairNLP/flair, last access: 14/03/2023.
5 HuggingFace Transformers, https://huggingface.co/docs, last access: 14/03/2023.

Table 1. Comparison between sentiment analysis tools.

Tool	Approach	Dataset	Scoring
TextBlob	Ruled-based	Customer/product reviews (mostly adjectives)[6]	From -1 to 1, where -1 means very negative, 1 means very positive and 0 means neutral.
Vader	Ruled-based	Multiple domains[7] (social media)	Same as TextBlob.
TextBlob + NaieveBayerAn-alyzer	Machine Learning (Naïve Bayes)	Movie reviews (NLTK)[8]	Returns a value of positivity and a value of negativity. Both in ranges between 0 and 1.
Flair	Machine Learning (Embedding-based models)	Movie reviews (IMDB)[9]	Returns a label (Positive or Negative) with a score ranging from 0 (uncertain) to 1 (very certain) about the prediction.
HuggingFace Transformers	Machine Learning (Deep Learning)	Stanford Sentiment Treebankmm-sst2[10]	Same as Flair.

3 Evaluating the performance of sentiment analysis tools

In this section, two evaluations are conducted to assess the effectiveness of the four tools presented in Section 2. Two domains were selected, taking into account the datasets with which the four tools were built: films and television and opinions on public figures. Two sets of tweets in English[11] associated with these domains were downloaded and cleaned following the guidelines provided in [7]. After that, the tweets were manually labeled by a human as positive (0.3 to 1), negative (-1 to -0.3), or neutral (-0.3 to 0.3). This process could cause biases and errors. However, it provides valuable ground truth for comparing the different sentiment analysis tools. This labeling process was essential for us to have a baseline of comparison for the sentiment analysis tools being tested and to evaluate their performance and accuracy in classifying sentiments within these domains. In order to gauge the level of efficiency of each tool, the obtained results are compared with what was manually determined; they are classified into one of the following six categories presented in Table 2.

[6] TextBlob Lexicon. https://github.com/sloria/TextBlob/blob/dev/textblob/en/en-sentiment .xml, last access: 14/03/2023.

[7] Vader Lexicon, https://github.com/cjhutto/vaderSentiment/blob/master/vaderSentiment/ vader_lexicon.txt, last access: 14/03/2023.

[8] NLTK Corpus Movie Reviews Dataset. https://www.kaggle.com/datasets/nltkdata/movie-review, last access: 14/03/2023.

[9] IMDB Large Movie Review Dataset. https://github.com/flairNLP/flair/blob/master/ tests/resources/ tasks/imdb/README.md, last access: 14/03/2023.

[10] The Stanford Sentiment Treebank dataset. https://huggingface.co/datasets/sst2 last access: 14/03/2023.

[11] Note that some tools lack support for languages other than English.

Table 2. Categories in which the results are classified.

Category	Condition
True positive *(TP)*	when it is positive and correctly predicted as positive.
True negative *(TN)*	when it is negative and correctly predicted as negative.
True neutral *(TNL)*	when it is neutral and correctly predicted as neutral.
False positive *(FP)*	when it is not positive and incorrectly predicted as positive.
False negative *(FN)*	when it is not negative and incorrectly predicted as negative.
False neutral *(FNL)*	when it is not neutral and incorrectly predicted as neutral.

The accuracy rates are calculated by dividing the total number of correctly classified tweets (TP + TN + TNL) by the total number of tweets evaluated.

The first selected domain to evaluate the performance of sentiment analysis tools when dealing with tweets was ‘Films and Television’. To accomplish this, we searched for tweets containing the term ‘Wakanda Forever’, corresponding to the latest Marvel movie release on Disney+. One hundred tweets were collected and then manually labeled using the before mention criteria. The dataset generated comprised 37 positive, 43 neutral, and 20 negative tweets. After that, we compared each tool's efficiency in classifying tweets according to their level of sentiment, as presented in Table 3. According to these results, TextBlob obtained the highest accuracy rate.

The second domain was 'Opinion on public figures'. We collected and manually categorized two hundred tweets with opinions about 'Elon Musk', who has been a controversial figure since he acquired Twitter; 46 were positive, 48 were neutral, and 106 were negative. As shown in Table 4, HuggingFace Transformers had the highest accuracy rate, followed by Flair.

Table 3. Results of the evaluation - Domain ‘Films and Television’.

Tool	TP	TN	TNL	FP	FN	FNL	Accuracy
TextBlob	12	6	36	9	1	36	54%
Vader	15	5	24	21	4	31	44%
TextBlob+NaieveBayerAnalyzer	19	6	23	17	10	25	48%
Flair	28	12	3	34	14	9	43%
HuggingFace Transformers	34	17	0	24	25	0	51%

Table 4. Results of the evaluation - Domain ‘Opinion on public figures’.

Tool	TP	TN	TNL	FP	FN	FNL	Accuracy
TextBlob	10	14	46	6	3	121	35%
Vader	20	42	44	23	4	67	53%
TextBlob+NaieveBayerAnalyzer	17	24	17	67	67	64	29%
Flair	29	88	3	24	39	17	60%
HuggingFace Transformers	33	97	0	25	45	0	65%

4 Conclusions

Four tools for sentiment analysis were compared to show how it is possible to use them without training a custom model. Each tool has its peculiarities; some were developed on a rule-based approach (TextBlob, Vader), while others use machine learning techniques (such as TextBlob+NaieveBayerAnalyzer, Flair, HuggingFace Transformers). Tables 3 and 4 allow observing that the approach used for each tool does not impact the accuracy. For example, TextBlob obtained 54% versus HuggingFace Transformers, with 51% in Table 3. Generally, machine learning techniques are considered less efficient because they assign word-by-word weighting without considering the message context [3]. Additionally, two tools that use the same approach may have widely different accuracies, such as TextBlob (35%) and Vader (53%) in Table 4.

On another side, some tools use domain-specific datasets, and others use multi-domain or general-purpose data. However, this is not a conditional aspect of the tool's accuracy. For example, for the first evaluation focused on film and television, TextBlob scored the highest accuracy (54%) in Table 3, despite using a lexicon from a different domain.

In both evaluations, the percentages of correct answers were relatively low, reaching a maximum of 65%. The manual labeling of tweets was subjective, and the datasets used for the evaluations were relatively small (100 and 200), so more evaluations are required to define a conclusion.

We expect this paper will contribute to discussing how practitioners or researchers could choose the correct sentiment analysis tools without requiring expert knowledge.

References

1. D'Andrea A., Ferri F., Grifoni P., Guzzo T.: Approaches, Tools and Applications for Sentiment Analysis Implementation. International Journal of Computer Applications 125(3), 26-33 (2016).
2. Rambocas M., Gama, J.: Marketing Research: The Role of sentiment Analysis. Universidade do Porto, Faculdade de Economia do Porto, paper 489 (2013).
3. Urmita M., Dhanraj V.: Sentiment Analysis of Facebook Using Textblob and Vader. Journal of Innovative Engineering and Research 4(1), 10-14 (2021).
4. Srivastava R., Bharti P.K., Verma P.: Comparative Analysis of Lexicon and Machine Learning Approach for Sentiment Analysis. International Journal of Advanced Computer Science and Applications 13(3) (2022).
5. Mahmood A., Kamaruddin S., Naser R., Mohd Nadzi M.: A Combination of Lexicon and Machine Learning Approaches for Sentiment Analysis on Facebook. Journal of System and Management Sciences 10(3), 140-150 (2020).
6. Ray P., Chakrabarti A.: A Mixed approach of Deep Learning method and Rule-Based method to improve Aspect Level Sentiment Analysis. Applied Computing and Informatics 18(1/2), 163-178 (2022).
7. Akbik A., Blythe D., Vollgraf R.: Contextual String Embeddings for Sequence Labeling. In: Proceedings of the 27th international conference on computational linguistics, pp. 1638-1649, Association for Computational Linguistics, USA (2018).

Early detection of grapevine diseases using pre-trained Convolutional Neural Networks

Cristian Emmanuel Rios[1], César A. Estrebou[2], and Fernando Emmanuel Frati[1]

[1] Department of Basic and Technological Sciences,
National University of Chilecito, Chilecito, La Rioja, Argentina
crios@undec.edu.ar, fefrati@undec.edu.ar

[2] Computer Science Research Institute LIDI, School of Computer Science, National University of La Plata, La Plata, Buenos Aires, Argentina
cesarest@lidi.info.unlp.edu.ar

Abstract. This paper proposes to apply pre-trained Convolutional Neural Networks (CNN) for the early detection of two common grapevine diseases: peronospora and oídio. These diseases present similar symptoms and are of great viticultural importance. Our objective is to train a CNN using transfer learning techniques to accurately detect the presence of early symptoms of the diseases under study. To achieve that, we'll design a pipeline that starts with data acquisition in the field and finalizes with the early disease identification, including class definition, labeling, image preprocessing and training process of the CNN, employing edge computing-based service computing paradigm to overcome some inherent problems of traditional mobile cloud computing paradigm.

Keywords: Deep Learning · Convolutional Neural Networks · Object Detection · Edge Computing · Inclusive Inteligent Systems

1 Introduction and related work

Field-grown grapevines are considered one of the main crops of La Rioja. Peronospora and Oídio are two similar diseases that frequently attack this crop. Peronospora is a sporadic but severe disease that directly affects production, while Oídio mainly attacks the grapevine leaves. As both diseases decreasing the quantity and quality of the bunches, them must be controlled. On the other hand, consumer market trends are rapidly expanding towards organic agriculture.

Consequently, accurate and early detection and identification of these diseases would improve the grapevine farmer's opportunities to reach export markets, applying phytosanitary products used to control them in fewer quantities and at the appropriate crop stages.

Our main objective is, under an inclusive approach, to design an intelligent system for the early detection of grapevine diseases to carry out to farmers who (frequently) are prevented from access to technology by different factors. To achieve that, we will design a complete edge-to-cloud system [1] that could be used in real-time in the field, keeping data and computation close to the end user. We propose to leverage farmer smartphone capabilities (low latency, reduced power consumption, and location awareness) to sense, image capture, and run the software that implements the pre-trained CNN for detection. On the other hand, we will deploy private cloud infrastructure on the University servers to supply the power computing needed for the remanent system parts.

We pretend to achieve that employing CNNs techniques, taking advantage of the fact that pre-trained models provide highly discriminative information, despite being trained on scenarios and tasks completely different. There are significant examples of the application of CNNs for image classification in several fields:

Breast cancer detection using histopathology images and pre-trained deep learning models. This study, carried out on with digital histopathology images, evaluates three scenarios, starting from a classical machine learning scheme and logistic regression combined with principal component analysis. Then they include the use of pre-trained deep models and finally a deep model based on a convolutional neural network [2].

Recognition of necrotic lesions for the detection of the thrips plagues in peas using the deep learning model yolov4-tiny. This research proposes a fast and effective necrotic lesion detection system for the early detection of the thrips infestation in peas by implementing the yolov4-tiny deep learning method[3].

Tomato diseases and pests detection based on improved Yolo V3 Convolutional Neural Network. This study builds the dataset of tomato diseases and pests under the real natural environment, optimizes the feature layer of the Yolo V3 model by using an image pyramid to achieve multi-scale feature detection, improves the detection accuracy and speed of Yolo V3 model, and detects the location and category of diseases and pests of tomato accurately and quickly [4].

A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure. CNN-based visual food recognition algorithms to achieve the best-in-class recognition accuracy, where edge computing was employed to overcome the system latency and low battery life of mobile devices. [5].

With this in mind, we propose to design a metodology that starts with data acquisition in the field and finalizes with the disease identification, including class definition, labeling, image preprocessing and CNN training process, using in its implementation, the Edge Computing paradigm.

2 Background

Peronospora is a disease that can penetrate the plant organs of the vine and develop inside the leaves, stems, and fruit, causing significant damage and consequences to the plant structure. Initially, it produces more or less circular, translucent, oily-looking spots on the leaves (Fig. 1.a).

Oídio, on the other hand, is a fungus that does not penetrate the plant organs but is found on the upper or lower side of the leaves. Initially, it produces spots similar to white or ashy powdery (Fig. 1.c). The nature of the spots they produce is one of the main differences between each disease. However, it is not easily distinguishable in such early stages. The differences between the two diseases become more evident as the signs progress. Fig. 1 b and Fig. 1 d show from left to right the progression of symptoms of peronospora and oídio in advanced stages, respectively [6].

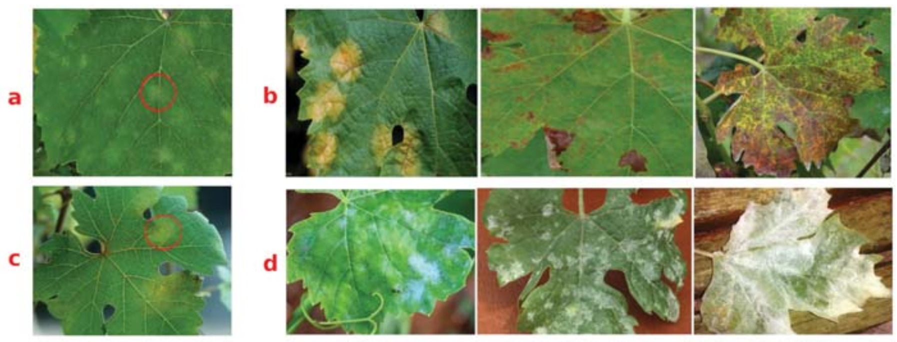


Fig. 1. a) Early symptom of peronospora, b) evolution of peronospora symptoms (from left to right), c) early symptom of oídio d) evolution of oídio symptoms (from left to right).

On the other hand, a new factor to be taken into account in agriculture is organic production, which is regulated in Argentina by Law 25.127 [7], required by an ever more demanding export market. To comply with this law, the farmer must avoid using synthetic fertilizers, pesticides, or genetic manipulation. It also promotes disease control biological methods and minimizes air, soil, and water pollution. Given the above, early detection of these diseases can impact significatly at the grapevine grower.

3 Proposal

Our proposal consist of the next stages:

Field data collection. The leaf of the plant is the plant organ that shows early symptoms of both Oídio and Peronospora. For this reason, we will work with images of leaves in their natural state and different light conditions using a mobile device connected to the Internet to facilitate rapid diagnosis. Images may have a complex background, but the leaf will take up most of the area. Although this could complicate the process of image analysis and, later, the training of the neural network, it is essential to make a diagnosis in the field as quickly as possible.

Class definition. Considering that it is more difficult to distinguish between Oídio and Peronospora in the early stages, experts who can recognize the evolution of symptoms in both diseases must supervise the composition of the image set. So, as we propose to use pre-trained CNN models, we will need to define classes for training. For each disease, we suggest classifying it in the early stages and advanced stages. Table 1 describes the characteristics that the subset of images belonging to each class should have.

Table 1. Description of classes

Nro	Name	Features
1	Early peronospora	Presence of more or less circular spots of a few centimeters, translucent and with the appearance of "oil stains".
2	Advanced peronospora	Presence of chlorotic spots that can reach several centimeters in diameter, visible on both sides of the leaf, with whitish efflorescence lesions on the underside. In adult leaves, the manifestation of mosaic lesions and/or necrosis.
3	Early oídio	Presence of more or less circular spots of a few centimeters, similar to white or ashy powder. Sometimes at the beginning of the attack, they appear as small oily spots on the upper side of the leaves, together with brownish dots.
4	Advanced oídio	Leaves appear twisted or curled, covered with dust on the upper and lower sides, and accompanied by necrosis.

Image labeling. Applying a label to all or part of an image to indicate the class to which it belongs is a necessary step in supervised learning. An expert in plant disease identification should carry out this process. The labeling method depends on the general approach chosen for image analysis. For example, if we want to classify them, we could move each image to different labeled folders or to integrate labels in the image metadata. Instead, if we want to detect objects, we should outline the region of interest (ROI) in the image, sketching the coordinates of the target object. Often, the ROI shape is rectangular but can fit more precisely to the target object.

Image preprocessing. The images must fit the size of the CNN input layer. If not, we need to resize them first. Colors are relevant in this case study (Oídio tends to be grayish white, while Peronospora tends to be oil stain-like shades). Therefore, we will not do a grayscale conversion. We will have to normalize the dataset to maintain fairness across all images and ensure they contribute equally to the loss function. So, we must scale all images to an equal range of [0,1]. Finally, we will use data augmentation to increase the diversity of the dataset without having to collect new images. This process consists of performing transformations on the existing image files (flipping, rotating, and changing brightness) that do not alter the identification of the diseases [8].

CNN transfer learning training. To find the best model for the case study, we propose to study the effect of model complexity on classification accuracy for four CNN models: AlexNet[9], GoogLeNet[10], ResNet[11], and YOLO[12].

4 Preliminary conclusions

The detailed analysis of fungal diseases for the case study allowed us to delimit temporally and spatially the evolution of Oídio and Peronospora on grapevine leaves. The review of related works shows us that pre-trained models could be efficient solutions for the early detection of the described symptoms. We proposed a methodology for data acquisition, training image classification, labeling, and preprocessing techniques for the case study that will result in the design of the edge-to-cloud computing architecture according to the requirements of early detection in the vineyard.

References

1. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on Mobile Edge Computing: The communication perspective, https://arxiv.org/abs/1701.01090.
2. Gaviria, H.A., Sarria-Paja, M.: Detección de Cáncer de Seno usando imágenes de histopatología y modelos de aprendizaje profundo pre-entrenados, https://revistascientificas.cuc.edu.co/CESTA/article/view/3922/3919.
3. Mosquera, S.D.M., Guerrero-Andrade, C.J.: Reconocimiento de lesiones Necróticas para la detección de la plaga thrips en el guisante mediante el uso del modelo deep learning Yolov4 Tiny, https://lajc.epn.edu.ec/index.php/LAJC/article/view/296.
4. Liu, J., Wang, X.: Tomato diseases and pests detection based on improved Yolo v3 convolutional neural network, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7309963/.
5. Liu C., Cao Y., Luo Y., Chen G., Vokkarane V., Yunsheng M., Chen S., Hou P.: A New Deep Learning-based Food Recognition System for Dietary Assessment on An Edge Computing Service Infrastructure, IEEE Transactions on Services Computing (2017).
6. Pizzuolo, P., Lucero, G.: Enfermedades Más frecuentes, que afectan a la canopia de la vid en la zona vitivinicola del centro-oeste de Argentina, Asociación Argentina de Fitopatólogos (2020).
7. https://www.argentina.gob.ar/senasa
8. Chollet, F.: Deep learning with python. Manning Publications, Shelter Islands (2018).
9. Inc, A.K.G., Krizhevsky, A., Inc, G., Profile, G.I.V., Inc, I.S.G., Sutskever, I., OpenAI, G.E.H., Hinton, G.E., OpenAI, Profile, O.A.I.V., Machinery, A.for C., Metrics, O.M.V.A.: ImageNet classification with deep convolutional Neural Networks, https://dl.acm.org/doi/10.1145/3065386.
10. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions, https://research.google/pubs/pub43022/.
11. Amaya Rodríguez, I., Civit Masot, J.,...,Brandao, P. (2021). ResNet. En J. Bernal, A. Histace (Ed.), Computer-Aided Analysis of Gastrointestinal Videos (pp. 99-114). Cham, Switzerland: Springer.
12. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection — IEEE ..., https://ieeexplore.ieee.org/document/7780460.

Use of Scalable Fuzzy Time Series Methods to predict Electrical Demand

José Miguel Rubio León$^{1[0000-0003-0377-4397]}$ and José Manuel Rubio Cienfuegos^{2}

^{1} Universidad Bernardo O'Higgins, Avda Viel 1497, Santiago de Chile, Chile
josemiguel.rubio@ubo.cl

^{2} Universidad de Chile Avda Beauchef 850, Santiago de Chile. Chile
jose.rubio@ug.uchile.cl

Abstract. Electric power is one of the main engines of humanity since the late eighteenth century, therefore, understanding the behavior of the demand of this corresponds to a very important study. Within this field, electricity demand prediction has been one of the most developed activities in several researches, where the use of fuzzy time series models has had good results. This paper presents the use of scalable fuzzy time series methods, which were developed by Petronio Silva [1] to predict the Spanish electricity demand collected by the Spanish TSO, whose computational implementation uses a Mamdani fuzzy inference system, which directly processes a time series containing the demand data and its register. It should be noted that the database also includes forecasts made by the same Spanish TSO, so comparisons are made with these forecasts.

Keywords: FTS methods · PWFTS model · Electrical demand

1 Fuzzy Time Series Models

Fuzzy time series (FTS) can refer to a time series composed of fuzzy linguistic terms, or to the family of nonparametric forecasting methods introduced by Song and Chissom [2] based on the fuzzy set theory of Zadeh [3]. These methods allow handling numerical and non-numerical data, and have been used to predict the university enrollments [4], stock markets [5], tourism [6], electric load [7][8], among other phenomena.

The scalable models developed by Silva are 2 consensus methods that incorporate the main developments of the FTS literature for methods based on high-order rules and weighted rules. He also proposes a new model that uses probabilistic predictions considering 2 sources of uncertainty: fuzzy measures and stochastic behavior, being this a single-variable prediction model, which is also used as part of a multivariable prediction system.

Single-variable models: Since fuzzy logic analyzes the degrees of membership of variables to fuzzy sets, a great deal of knowledge can be extracted by studying the relationships that time series values have with the association to different fuzzy sets. The simplest models that perform this work by analyzing only the values belonging to the time series are the single-variable fuzzy models.

- **HOFTS** (*High Order Fuzzy Time Series*): This model, whose main parameter for the creation of rules is the order, does not weight the partition or the fuzzy sets, assuming that each of them has the same importance.
- **WHOFTS** (*Weighted High Order Fuzzy Time Series*): System that generates rules from the order and importance of fuzzy sets.
- **PWFTS** (*Probabilistc Weighted Fuzzy Time Series*): Model that in addition to considering the analysis of the previous models for training, uses the rules generated to change the importance of the fuzzy sets in a probabilistic way, which is applied again to the rules, feeding them back with new inferences.

Multivariate Models: Fuzzy models can also study the relationships between the values of a time series with related variables. However, these extra variables must have their own partition, since they correspond to values associated with a different phenomenom.

- **MVFTS** (*MultiVariable Fuzzy Time Series*): System that analyzes the relationships between variables using the linguistic variables of each fuzzy time series assuming that each fuzzy set has the same importance for each variable. It is similar to the HOFTS models since it only uses the linguistic terms to perform its analysis, but with the ability to have a more robust analysis when obtaining a prediction since it also uses information related to the target variable.
- **WMVFTS** (*Weighted Multivariable Fuzzy Time Series*: Extension of the previous system, where it adds importance to fuzzy sets.
- **FIG-FTS** (*Fuzzy Information Granular Fuzzy Time Series*): Model that has an emphasis on the relationships between variables through the use of a k-nearest neighbors system, in addition to incorporating weights between fuzzy sets and rules generated using the PWFTS system.

2 Dataset description

The information used for testing the different prediction models presented in this article is data collected by ENTSO-E (European Network of Transmission System Operators for Electricity) [10], a body of electricity managers that contains records on various variables related to the electricity market of several European countries. The main information in this database corresponds to the hourly Spanish electricity demand between 2015 and 2019 together with its date, the time of registration in format `yyyy/mm/dd - hh:mm:ss` and the forecast of electrical demand given by the organization itself. This dataset has 35064 samples, and for this work the sample characteristics used were those mentioned above, and was processed as a time series before being taken to the scalable FTS models.

3 Metodology and Implementation

The fuzzy logic systems used in this work are the scalable FTS models those described in section 1, developed by Petrônio Cândido de Lima e Silva in the `python pyFTS` [11] library, using Google Collaboratory environment to make the programation fo the systems. The programming was performed in Google Colaboratory with Nvidia 530.30.02 drivers and Python 3.9.16. . Given this, the computational process when using a fuzzy model use a Mamdani fuzzy inference system and therefore used the following procedure for training.

1. **Data preprocessing:** This is a generic step when using prediction models. Here all the cleaning and filtering of the information is done to take it to the models.
2. **Partition configuration:** This is the most important step when using fuzzy models, a precise analysis is performed on the target variable to know which is the most appropriate type of partition and number of fuzzy sets. The type of model to be used and its order is also configured.
3. **Fuzzyfication of the data:** The data are transferred to the fuzzy domain. At this point the fuzzy time series are generated.
4. **Generation of fuzzy rules:** Here the temporal transition rules are obtained, where they depend on the parameters with which the partition was configured. Depending on the model, the importance of the fuzzy sets (weighting models) and the relationships between the processed variables (multivariable models) are obtained.
5. **Forecast:** The predictions are made by taking a series of values whose length must be at least the order configured in the partition, in order to be able to make the prediction for the following periods.
6. **Defuzzyfication of the data:** Process opposite to fuzzyfication, at this point the data is returned to its original domain.

The data were cleaned of empty samples and since we were working with hourly electricity demand, **5 levels** were configured: Very Low, Low, Medium, High and Very High, where these sets were in turn divided into 7 sub levels, and the partition used was the grid with triangular membership function, giving the same range of membership to each fuzzy set depending on the value of the time series.

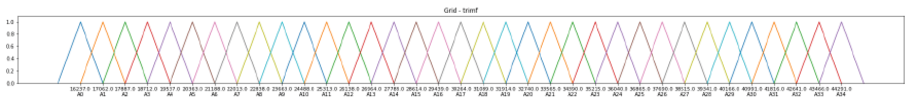


Fig. 1: Partition used for the data, 35 fuzzy sets were created with a minimum membership value of 15412 MW and a maximum of 45116, each interval had a length of 1650 MW.

The model was set up to apply this partition to the demand time series data, without using time windows, so that the series were processed all at once.

When multivariable models were used, the time variable was incorporated, and these series had their own partition. A lattice partition was also applied, but the records were processed by month and time, generating 2 fuzzy series with 12 and 24 fuzzy sets respectively, which were configured separately from the model. It should be noted that the partition applied to the demand data was the same as the one used in the single-variable models, except that it was not configured in the model, but together with the time partitions.

4 Results.

The predictions obtained by the single-variable fuzzy models are shown in Figure 2, where the shape of the forecasts obtained was similar over the entire time series of electricity demand. On the other hand, the multivariate fuzzy models had the behavior shown in Figure 3, where, as in the previous case, similar predictions were also obtained in other demand intervals.

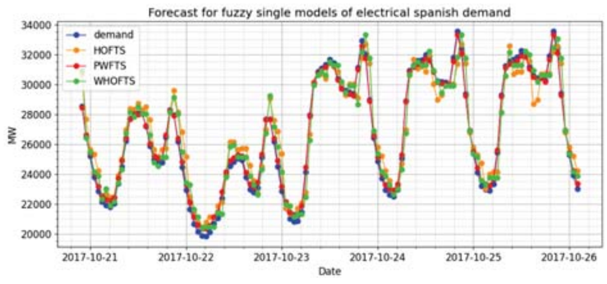


Fig. 2: Spanish electricity demand forecasts by fuzzy single variable models

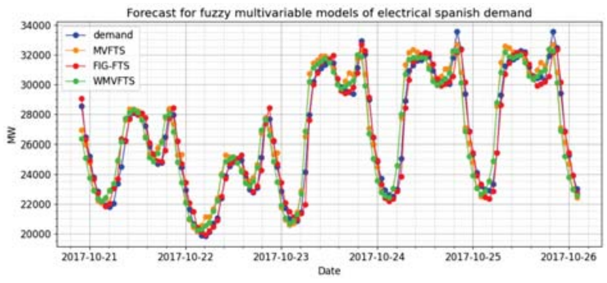


Fig. 3: Spanish electricity demand forecasts by fuzzy multivariable systems

The MAPE (Mean Absolute Percentage Error) was used as an error metric to evaluate the performance of the different systems.

Table 1: Performance of fuzzy multivariable prediction models

Model	Type	Order	MAPE
PWFTS	Fuzzy single-variable	1	0.8738
TSO Forecast	Not Applicable	Not Applicable	0.9590
FIG-FTS (k=2)	Fuzzy Multivariable	2	1.5421
FIG-FTS (k=3)	Fuzzy Multivariable	2	1.5866
WHOFTS	Fuzzy single-variable	3	2.3359
WHOFTS	Fuzzy single-variable	2	2.5408
HOFTS	Fuzzy single-variable	3	2.9445
WMVFTS	Fuzzy Multivariable	2	3.4820
MVFTS	Fuzzy Multivariable	1	3.6086
HOFTS	Fuzzy single-variable	2	3.6125

In general, the models were able to perform well in the one-step predictions, where the fuzzy single-variable systems did the job better and worse. This is partly due to the fact that although all of them have within their computational implementation the fuzzy logic theory, there are notorious differences in the generation of rules. Therefore, the values of the linguistic variable may differ quite a lot, which translates into different performance.

The fuzzy models that did not weight the sets were the ones that had a lower performance. This shows the great importance of this feature when setting up fuzzy models in general, since we will have more relevant fuzzy sets when generating the fuzzy rules and making the inferences, and later to make the demand predictions.

It should be noted that the single-variable models performed better if they were configured with a higher order, since the generation of their rules is strengthened if they have a larger amount of data to perform the analysis. However this does not happen with the PWFTS model, because this system generates its rules probabilistically, using the previous rules already generated to improve its analysis, so if a larger amount of samples are taken for the generation of a current rule then it reduces the amount of rules generated during its training, which leads to a reduction in its performance.

Unfortunately, it was not possible to find a description for the prediction method used by the Spanish TSO

5 Conclusion

The scalable models were able to make good one-step predictions from the demand data collected by the Spanish TSO. Although there were also configurations of these models that presented appreciable errors when forecasting demand such as the MVFTS and HOFTS models. However, these performances can be improved by changing the values of their hyperparameters, such as order.

The PWFTS model outperformed the predictions made by the Spanish TSO, thus showing the effectiveness of this system for time series predictions; however, this model is still under development and revision, so the results of the predictions obtained if this system is used with other databases should be carefully analyzed.

Although the use of this technique is not very well known, the work developed showed that it is very useful to work as prediction models. However, in order to demonstrate that these models can be the best for energy demand predictions, it is necessary to test them also with other databases.

References

1. P. Silva, *Scalable Models for probabilistic forecastng with fuzzy time series* Belo Horizonte - Minas Gerais Federal University of Minas Gerais (2019)
2. Q. Song and B. S. Chissom. Fuzzy time series and its models. *Fuzzy Sets and Systems*, 54 (3), pp:269-277, 1993. doi: 10.1016/0165-0114(93)90372-O.
3. L. A. Zadeh. Fuzzy Sets. *Information and Control* 8:338–353, 1965. DOI: 10.1016/S0019-9958(65)90241-X.
4. Q. Song and B. S. Chissorn. Forecasting enrollments with fuzzy time series-part II. *Fuzzy Sets and Systems* 62:1–8, 1994. doi: 10.1016/0165-0114(94)90067-1
5. M.-Y. Chen. A high-order fuzzy time series forecasting model for internet stock trading. *Future Generation Computer Systems*, 37:461–467, 2014. doi: 10.1016/j.future.2013.09.025.
6. M. H. Lee and H. Javedani. A Weighted Fuzzy Integrated Time Series for Forecasting Tourist Arrivals. In *International Conference on Informatics Engineering and Information Science* pp:206-217, Berlin Heidelberg, 2011. Springer. doi: 10.1007/978-3-642-25453-6_19
7. Z. Ismail, R. Efendi, and M. M. Deris. Application of Fuzzy Time Series Approach in Electric Load Forecasting. *New Mathematics and Natural Computation* 11(3), pp:229-248, 2015. doi:10.1142/S1793005715500076.
8. H. J. Sadaei, F. G. Guimarães, C. J. d. Silva, M. H. Lee, and T. Eslami. Short-term load forecasting method based on fuzzy time series, seasonality and long memory process. *International Journal of Approximate Reasoning*, 83:196–217, 2017. doi: 10.1016/j.ijar.2017.01.006.
9. S. Chen, *Forecasting enrollments based on fuzzy time series*, vol. 81, no. 3, pp. 311-319. 1996
10. N. Jhana, *Hourly energy demand generation and weather*, https://bit.ly/3nfphuw. Updated 2020
11. M. Intelligence and D. S. Laboratory, *pyFTS Quick Start*. 2018

Software Engineering and Databases

Integrating Oceanographic Sensor Data Using SSN/SOSA Ontology

Gustavo Nuñez[1,⋆], Carlos Buckle[1][0000−0003−0722−0949], and Marcos Zárate[1,2][0000−0001−8851−8602]

[1] Laboratorio de Investigaciones en Informática, Facultad de Ingeniería, Universidad Nacional de la Patagonia San Juan Bosco (LINVI-UNPSJB), Puerto Madryn, Argentina.
gnunez@ing.unp.edu.arcbuckle@unpata.edu.ar
[2] Centre for the Study of Marine Systems, Patagonian National Research Centre (CESIMAR-CENPAT-CONICET), Puerto Madryn, Argentina.
zarate@cenpat-conicet.gob.ar

Abstract As the deployment of ocean sensors continues to grow, there is a growing need for standardized ways to represent and integrate sensor data from different sources. One approach to achieving this is through the use of Semantic Sensor Network (SSN/SOSA) ontology, which provides a common vocabulary and framework for describing sensors, observations, and their properties. In this paper, we present a method for converting ocean sensor data in CSV format to Resource Description Framework (RDF) using RDFLib library for Phyton and SSN/SOSA ontology. The resulting RDF triples can be stored in a triplestore for querying and analysis, providing a standardized representation of ocean sensor data that can be easily integrated with other data sources.

Keywords: Ocean sensors · SSN/SOSA ontology · RDFLib · Aqualink.

1 Introduction and motivation

The ocean is a complex and dynamic system that plays a critical role in the Earth's climate and ecosystem [1]. Understanding the ocean requires the collection and analysis of vast amounts of data from a variety of sources, including oceanographic sensors. However, integrating and interpreting these data can be challenging due to the diverse nature of sensors and the lack of standardized approaches for describing them. The use of Semantic Sensor Network (SSN/SOSA) ontology [2] in ocean sciences has gained significant attention in recent years. This ontology provides a standardized framework for describing sensors, observations, and their metadata, allowing for seamless integration and interoperability of data from various sources. SSN/SOSA is designed to enable machine-readable descriptions of sensors, their capabilities, and the characteristics of the observations they make, which can be used to facilitate data discovery, integration, and interpretation.

⋆ Corresponding author.

In ocean sciences has been applied to a wide range of applications, including oceanographic monitoring, marine biology, and climate science. For example, in oceanographic monitoring, SSN/SOSA have been used to describe the properties of oceanographic sensors, such as their measurement range, accuracy, and resolution, as well as their deployment location and time series data. This has allowed for the integration of data from various sensors, including ocean gliders, buoys, and satellites, to better understand oceanographic processes and phenomena.

In climate science have been used to describe sensors used to monitor environmental parameters that affect the Earth's climate, such as temperature, salinity, and sea level. This has allowed for the integration of data from various sources, such as ships, buoys, and satellites, to better understand the Earth's climate system and its response to changes in the environment. Despite the potential benefits of using SSN/SOSA in ocean sciences, there are still challenges that need to be addressed. One of the main challenges is the lack of a standardized approach for implementing SSN/SOSA, which can lead to inconsistencies in the way sensors and observations are described. Additionally, there is a need for improved data management and data sharing practices to ensure that SSN/SOSA data is easily accessible and can be used by a wide range of stakeholders. Overall, the use of SSN/SOSA in ocean sciences has the potential to significantly improve our understanding of the oceans and their role in the Earth's system. As such, there is a need for continued efforts to develop and promote the use of SSN/SOSA in ocean science research and applications.

In this short paper, we present a method for converting ocean sensor data in CSV format extracted from Aqualink project to RDF using RDFLib [3] and SSN/SOSA ontology. A set of queries is also provided to retrieve the information from the resulting graph. The remainder of this paper is structured as follows: Section 2 presents the most relevant characteristics of the sensors used to create the graph. Section 3 shows the steps needed to convert to RDF using RDFLib and a set of queries that demonstrate its use. Finally, in Section 4, we present some conclusions related to previous experiences.

2 Background

The Aqualink Monitoring for marine ecosystems project[3] is a collaborative effort between multiple research institutions and government agencies aimed at developing and deploying an advanced monitoring system for marine ecosystems. The project is focused on the development of a standardized sensor network infrastructure that can be used to collect, integrate, and disseminate data from a wide variety of sensors, including those measuring physical, chemical, and biological parameters.

To carry out the experiments, information provided by the sensors that belong to the oceanographic buoy located in Isla Leones, Argentina (LAT: -45.043 LONG: -65.607)[4]. The station that is equipped with a suite of sensors, including

[3] https://aqualink.org/

[4] https://aqualink.org/sites/1136

those for measuring water temperature, wind speed and significant wave height. Figure 1 shows the schematic diagram of the buoy and the sensors associated with it. The data collected by these sensors is transmitted in near real-time to a central data management system, where it is quality-checked, processed, and made available to researchers and resource managers through a web-based portal. The data collected by the station can be used to inform a range of research and management activities, including the assessment of marine biodiversity, the detection of harmful algal blooms, and the monitoring of oceanographic processes such as up-welling and water column stratification.

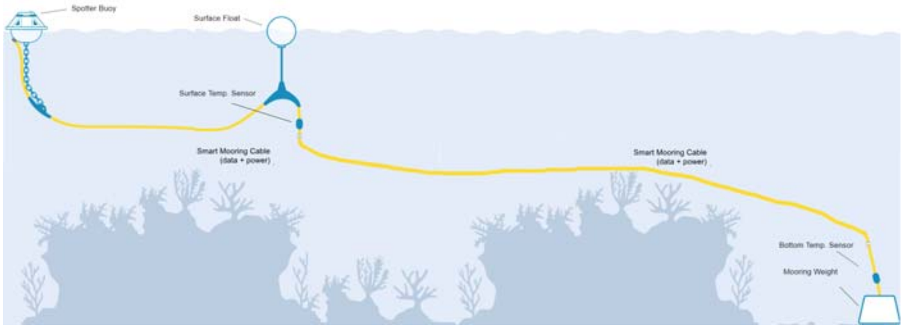


Fig. 1. Schematic diagram of the buoy and its sensors provided by Aqualink.

3 Case study

To illustrate the use of SSN/SOSA ontology in ocean sciences, a case study of Aqualink initiative is presented. The oceanographic variables that will be used to create the graph are the following: top temperature, bottom temperature, wind speed and significant wave height. The range of dates used goes from January 24, 2023 to February 24, 2023 with a total of 2418 measurements of the variables mentioned above.

We mapped the sensor properties and metadata to SSN/SOSA ontology classes and properties, including the following: `sosa:Platform`: represents a physical or virtual entity that carries one or more sensors and can act as a host for observing and measuring the environment. `sosa:Sensor`: represents a physical device that can measure, observe, or estimate properties of the environment. `sosa:FeatureOfInterest`: represents the entity or phenomenon being observed or measured by a sensor. It can be a point or an area in space, a sample or a specimen, or an event or a process. `sosa:Observation`: represents a record of an act of observing or measuring a property of a feature of interest by a sensor. It captures the values of one or more observable properties at a particular time and place. `sosa:Result`: represents the value or values of an observable property resulting from an observation and `sosa:ObservableProperty`: represents a property or phenomenon that can be observed or measured by a sensor. It can be a physical, chemical, biological, or environmental property, among others.

To model temporal entities the W3C time ontology [4] was used, to spatial entities we chose the GeoSPARQL ontology [5] and to model measurement units the Quantities, Units, Dimensions, and Types (QUDT) Ontology [6].

Figure 2 shows how station in Punta Leones was modeled. It can be seen that the buoy is of type `sosa:Platform`, the temperature sensor is of type `sosa:Sensor`, the surface temperature observations are of type `sosa:Observation`. Observation results are of type `sosa:Result` and `qudt:QuantityValue`.

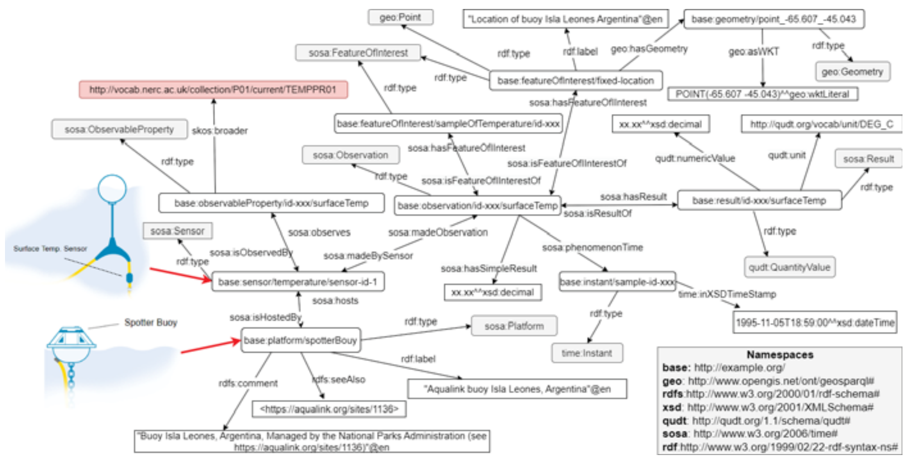

geo:Point
"Location of buoy Isla Leones Argentina"@en
base:geometry/point_-65.607_-45.043
sosa:FeatureOfInterest
rdf:type
rdf:label
geo:hasGeometry
geo:asWKT
rdf:type
geo:Geometry
rdf:type
base:featureOfInterest/fixed-location
http://vocab.nerc.ac.uk/collection/P01/current/TEMPPR01
rdf:type
POINT(-65.607 -45.043)^^geo:wktLiteral
sosa:hasFeatureOfInterest
sosa:ObservableProperty
skos:broader
base:featureOfInterest/sampleOfTemperature/id-xxx
xx.xx^^xsd:decimal
http://qudt.org/vocab/unit/DEG_C
rdf:type
sosa:Observation
sosa:hasFeatureOfInterest
sosa:isFeatureOfInterestOf
qudt:numericValue
qudt:unit
sosa:Result
rdf:type
sosa:isFeatureOfInterestOf
rdf:type
base:observableProperty/id-xxx/surfaceTemp
base:observation/id-xxx/surfaceTemp
sosa:hasResult
sosa:isResultOf
base:result/id-xxx/surfaceTemp
sosa:Sensor
sosa:observes
sosa:madeObservation
rdf:type
Surface Temp. Sensor
rdf:type
sosa:isObservedBy
sosa:madeBySensor
sosa:phenomenonTime
qudt:QuantityValue
sosa:hasSimpleResult
base:sensor/temperature/sensor-id-1
base:instant/sample-id-xxx
xx.xx^^xsd:decimal
time:inXSDTimeStamp
sosa:hosts
rdf:type
Spotter Buoy
sosa:isHostedBy
rdf:type
sosa:Platform
1995-11-05T18:59:00^^xsd:dateTime
base:platform/spotterBouy
time:Instant
rdf:label
rdfs:comment
rdfs:seeAlso
"Aqualink buoy Isla Leones, Argentina"@en
<https://aqualink.org/sites/1136>
"Buoy Isla Leones, Argentina, Managed by the National Parks Administration (see https://aqualink.org/sites/1136)"@en
Namespaces
base: http://example.org/
geo: http://www.opengis.net/ont/geosparql#
rdfs:http://www.w3.org/2000/01/rdf-schema#
xsd: http://www.w3.org/2001/XMLSchema#
qudt: http://qudt.org/1.1/schema/qudt#
sosa: http://www.w3.org/2006/time#
rdf:http://www.w3.org/1999/02/22-rdf-syntax-ns#

Fig. 2. Simplified diagram of the resulting graph, in the image you can see the main classes and their relationships. For simplicity the diagram only shows the modeling of the sensor that records the surface temperature..

To facilitate the integration and interoperability of data from Aqualink sensors, we converted data in CSV format to RDF using RDFlib. The approach used consists of the following steps:

1. Load CSV data into DataFrame.
2. Create RDF graph with SSN/SOSA ontology and complementary ontologies.
3. Map CSV columns to RDF properties according to Figure 2 using RDFLib.
4. Save RDF graph to file.
5. Query the graph.

Google colab was used to perform all mappings, the code can be tested in the following link. The resulting RDF data can be easily queried and integrated with other RDF data sources, enabling more comprehensive analysis of oceanographic data.

3.1 Quering with SPARQL

RDFLib can create and manipulate RDF graphs and can be used to extract information from these graphs using SPARQL query language [7].

In this case study, it is necessary to obtain surface and bottom temperatures recorded during the date range from 2023-01-28 to 2023-01-30. To do this, it is first necessary to create an RDF graph, prepare a SPARQL query that selects all triples in the graph whose type is `sosa:Observation`, and execute the query using the query() method of the Graph object. The results of the query are returned as rows, which we can iterate over and print. To see the complete listo of queries developed see the following link to google colab.

4 Conclusions

The case study presented in this paper highlights the successful implementation of SSN/SOSA in the Aqualink project, which has enabled the collection and sharing of high-quality data on a range of physical, chemical, and biological parameters in marine environments. This has the potential to improve our understanding of marine ecosystems and inform more effective management practices. Overall, the use of SSN/SOSA ontologies in oceanographic sensor data sharing and collaboration has the potential to drive significant advancements in our understanding of the ocean and the impacts of human activities on marine ecosystems. Continued efforts to develop and apply these standards will be critical in achieving more effective management and conservation of our oceans.

References

1. Elvira S Poloczanska, Christopher J Brown, William J Sydeman, Wolfgang Kiessling, David S Schoeman, Pippa J Moore, Keith Brander, John F Bruno, Lauren B Buckley, Michael T Burrows, et al. Global imprint of climate change on marine life. *Nature Climate Change*, 3(10):919–925, 2013.
2. Armin Haller, Krzysztof Janowicz, Simon JD Cox, Maxime Lefrançois, Kerry Taylor, Danh Le Phuoc, Joshua Lieberman, Raúl García-Castro, Rob Atkinson, and Claus Stadler. The modular ssn ontology: A joint w3c and ogc standard specifying the semantics of sensors, observations, sampling, and actuation. *Semantic Web*, 10(1):9–32, 2019.
3. Carl Boettiger. *rdflib: A high level wrapper around the redland package for common rdf applications*, 2018.
4. Simon Cox and Chris Little. Time ontology in owl, w3c candidate recommendation 26 march 2020. [Online; accessed 14-Apr-2023].
5. Matthew Perry and John Herring. Ogc geosparql-a geographic query language for rdf data. *OGC implementation standard*, 2012.
6. Ralph Hodgson, Paul J Keller, Jack Hodges, and Jack Spivak. Qudt-quantities, units, dimensions and data types ontologies, 2014. [Online; accessed 14-Apr-2023].
7. Sparql 1.1 query language w3c recommendation 21 march 2013, 2013. [Online; accessed 14-Apr-2023].

NoSQL Database Design Processes - A Comparative Study

Luciano Marrero[1], Verena Olsowy [1], Pablo Thomas[1]

[1] Instituto de Investigación en Informática LIDI
Facultad de Informática - Universidad Nacional de La Plata – Argentina
Centro Asociado a la Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC)
526 e/ 10 y 11 La Plata Buenos Aires
{lmarrero, volsowy, pthomas}@lidi.info.unlp.edu.ar

Abstract. NoSQL databases have emerged as a response to scalability problems presented by relational databases when used in Big Data contexts. These databases do not have a standard process for designing data schemas, but have emerged as solutions directly at the physical level. As NoSQL databases have gained popularity, different NoSQL design processes and/or methodologies have been proposed, which are necessary to understand the semantics of the stored data. This paper presents a comparative study of NoSQL database design processes.

Keywords: NoSQL, Design NoSQL, Database Design Processes.

1 Introduction

For several years now, there have been software applications that have pushed relational databases to their performance limits. Examples of these applications are social networks, such as Twitter or Facebook, online shopping applications, such as Amazon or eBay; Internet of Things (IoT) applications, Smart Cities; or the use of Big Data, among others. These applications need to manage large volumes of data that are often distributed across multiple servers, must ensure adequate response times and high availability in contexts of a high number of concurrent requests. In these scenarios, relational databases have shown different scalability problems. In response to this problem, a new generation of database management systems, known as NoSQL, has emerged.

NoSQL is not only an alternative to relational databases, but is an umbrella term for various strategies for storing unstructured data. Initially, these databases emerged at the implementation level (physical level), and consequently without a defined process for their design [5, 6].

Traditional relational database design and construction methodologies have been extensively studied, applied and refined for decades. However, the principles and/or rules that apply to a relational data model are not appropriate for a NoSQL database. This is because they have a different implementation, therefore, their design process must also be different.

This paper proposes a comparative analysis of NoSQL database design processes.

Starting from section 2, the paper is organized as follows: first, the essential aspects of NoSQL databases are presented, section 3 presents the selected design processes, section 4 presents a comparative analysis, and finally, section 5 expresses conclusions and future work.

2 Main features of NoSQL databases

NoSQL differs from traditional relational database management systems in several respects; it does not have a structured query language (SQL) as its core language, does not require a fixed, tabular structure, does not support JOIN operations, does not fully guarantee ACID properties (atomicity, consistency, isolation and durability), and is generally suitable for horizontal scalability [6, 7, 8].

NoSQL proposes a system called "BASE (Basically Available, Soft State, Eventual Consistency)". Through these properties basic availability is achieved, meaning that the system will be available most of the time. With soft state the system becomes more flexible in terms of consistency and with eventual consistency it is guaranteed that the system will eventually become consistent [6, 7, 8]. There are four main types of storage for NoSQL Databases.

Key/Value Storage: Simple in implementation, they store data as a set of "key/value" pairs. The key represents a unique identifier that can return an arbitrary complex object of information, called a value. For example, Redis and DynamoDB, among others, implement this type of storage. [13, 14].

Documentary Storage: the central concept of this type of storage is the document. A Documental NoSQL Database stores, retrieves and manages documents. These documents encapsulate and encode data in some standard format (XML, YAML, JSON, BSON). For example, MongoDB and CouchDB, among others, are implementations of Document Databases.[9][10]

Column Family Storage: In this type of storage, data is organized by columns, rather than rows. For example, Cassandra and HBase, among others, use this type of storage. [15][16].

Graph Storage: the database is represented under the concept of a graph, allowing the information to be stored as nodes and their respective relationships with other nodes by means of edges. Graph theory is applied to traverse the structure. They are useful for storing information in contexts where there are numerous relationships between their data. Neo4j and OrientDB, among others, implement this type of storage. [11][12].

3 Design processes for NoSQL databases

In recent years, methodologies for the design of NoSQL databases have emerged. In [5] a literature review was conducted on this topic. As a result of this review, only three processes for the design of NoSQL databases have been identified [1], [2], [3, 4], which are analyzed in the present work.

3.1 Mortadelo: Automatic generation of NoSQL stores from plataform-independent data models [1]

Mortadelo is described as a model-based NoSQL database design process, where, starting from a conceptual data model, independent of any database type, a possible implementation for a specific NoSQL database system is autonomously generated. In addition, this process allows the final design to be configured according to the needs of each context.

This process needs to operate with well-defined models, in particular, it needs as input a metamodel in which the conceptual data model and the queries that will retrieve and update the represented information are specified. In addition, it allows certain general annotations to be made, e.g. the update rate that an entity is likely to have. This metamodel is called Generic Data Metamodel (GDM) and describes its components (entities, relations or references and queries) by means of a textual notation that has its own syntax.

In this process, starting from a case study, represented through a GDM, logical models are proposed for two categories of unstructured data storage, column family, where the Cassandra database engine [15] is used, and documentary, where the MongoDB database engine [9] is used. For this purpose, a set of predefined rules and algorithms are applied to transform an instance of a conceptual model into a concrete logical model for a NoSQL database type.

Finally, using another set of rules, they generate concrete code transformations for two database engines, Cassandra (column family storage) and MongoDB (document storage), i.e. the physical schema for a specific database engine is generated.

3.2 NoAM (NoSQL Abstract Model): Data Modeling in the NoSQL World [2]

A design process is proposed that has a conceptual phase, a logical phase, which is independent of the database type, and a final phase that considers the specific characteristics of a NoSQL database engine. NoAM is based on the following main activities:

A. Conceptual data modeling from Domain Driven Design (DDD) resulting in a UML diagram. There is no mention of how to realize this diagram.
B. On the UML diagram of the previous point, aggregates are identified. An aggregate is a group of related objects, representing an atomic unit of access and manipulation.
C. Implementation of the NoAM model based on the identification of aggregates.

The process begins with database design, building a conceptual representation of the data of interest, in terms of entities, relationships and attributes. Next, aggregations are identified. This activity may be driven by data access patterns, as well as by scalability and consistency needs. Specifically, aggregates must be designed as the units in which atomicity must be guaranteed. Each aggregate should include all the data required by a relevant data access operation. On the other hand, aggregates should be as small as possible. Small aggregates reduce concurrency collisions and meet performance and scalability requirements.

In this approach, NoAM is used as an intermediate model between aggregates and NoSQL databases. In NoAM, the unit of data access and distribution is modeled by a block, which represents a maximal unit of data for which atomic, efficient and scalable access operations are provided. NoSQL systems provide efficient, scalable and consistent operations on blocks and, in turn, this choice propagates such qualities to operations on aggregates.

Finally, we discuss how a NoAM data representation can be implemented in a specific NoSQL database engine (e.g., MongoDB [9]).

3.3 Modeling NoSQL databases: From Conceptual to Logical Level Design [3,4]

This approach proposes a common conceptual level model for several types of NoSQL databases and a NoSQL data specification language to represent a logical level data model, independent of any physical level representation. In addition, different validation rules have been proposed with respect to the conceptual model through the evolution of a case study.

This conceptual model has a common set of constructs, relationships and a set of meaningful properties of relationships to unify the conceptual level representations of different NoSQL databases. This model consists of three interrelated layers: Collection, Family and Attribute. The Attribute layer is the base layer of the conceptual model and the AT construct types are groups of all possible same instance types and elementary in nature.

The family layer is the middle layer of the conceptual model and can contain numerous types of FA constructs. This layer can be decomposed into multiple levels according to the designers' preferences.

The collection layer is the top layer of the conceptual model. The semantically related families of the top layer are assembled to form a column.

From a higher level, the database can be viewed as a set of columns.

The constructs of this model are connected to each other by distinct relationships. These relationships can be of two types: type relationship between layers and type relationship within the layer. These relationships have several properties, such as multiplicity, order, modality, availability, conditional participation and consistency.

A specification language is proposed to transform a conceptual data model into a logical model and then into a corresponding physical model for a NoSQL database engine.

Finally, a set of validation rules is proposed for the NoSQL model obtained, these rules are divided into three groups: for structural validation, for constraint validation and for consistency validation.

4 Comparative analysis

The three works present a design process for NoSQL databases. These three processes propose an approach which starts with a conceptual modeling stage, continues with a logical model and culminates with a physical model specific to a particular NoSQL database engine.

All three processes present details that should be taken into account when using and/or applying each of them.

In [1], it is required to define a metamodel called GDM that integrates a high-level conceptual model and the queries that will impact the database. Defining queries at an early stage can be prone to major changes at later stages when there are changing requirements. In addition, examples are presented for specific NoSQL databases, Cassandra for columnar family and MongoDB for documents. In the case of NoSQL databases with key-value storage, the possibility of applying this design process following the same rules as for column-family and document NoSQL databases is mentioned. For problems where the complete conceptual model and the main queries that will impact the database are available, it is a NoSQL database design process through which a physical model can be obtained for two types of NoSQL database engines, column-family (Cassandra) and document (MongoDB). For key-value NoSQL database types it is mentioned that it is an applicable process, following the same guidelines as for column-family and document engines, but no example is specified and with respect to graph-oriented NoSQL database types, no details of its application have been provided.

In [2], the starting point is a UML diagram [x] that is generated from a DDD (Domain-Driven Design) of which no details are given as to how it was done. On the UML model generated, the design of aggregates is carried out. This design requires knowledge of how the data will be retrieved and modified, which makes the process difficult, as it is generally difficult to establish the exact way in which the data will be manipulated at an early stage of the design. However, for documentary NoSQL database types, which have flexible schemas, or for key-value NoSQL databases, where the key must be clearly defined and is the only way to access the data, this design process can be adequately applied. For column-family NoSQL databases, e.g. Cassandra and/or graph-oriented databases, e.g. Neo4j [11], there is not enough detail to draw a conclusion.

In [3, 4] the design of a conceptual model specifically created for the proposed approach that has a set of layers or phases is required. In this conceptual model the existing relationships between the data is not explicit, but is at the layer level, something that can make it difficult to read and interpret. Subsequently, a logical and physical model is defined based on process-specific templates and finally a set of 18 rules that can be applied to obtain a final physical model must be analyzed. Compared to [1] and [2] this approach has formalisms and technicalities that make it a design process that demands a lot of attention and discipline from the designer. However, it provides an overview that supports changes dynamically and has important features with respect to data availability and replication, something that is not clear in [1] and [2].

In summary, in the case of using a NoSQL database engine that implements column family storage (e.g. Cassandra) or document storage (e.g. MongoDB), the design process proposed in [1] could be considered, this is because, in the examples presented, it is a complete process in its definition and its implementation is clearer and simpler compared to [2, 3, 4].

For graph-oriented NoSQL database engines, the design process defined in [3, 4] could be used, as it is the only one that considers this type of storage. For key-value NoSQL database engines, the design process in [1] could be used, if it is possible to generate the GDM metamodel. In the case of having only the conceptual model, one could consider the design process [2] or [3, 4] if one wants to use the phased conceptual model defined in that approach.

5 Conclusions and future work

This paper focuses on the analysis of three processes for the design of NoSQL databases. A literature review was conducted in [5], which identifies three design processes or methodologies that are presented or classified as applicable to more than one type of NoSQL databases [1], [2], [3, 4].

First, Mortadelo: *Automatic generation of NoSQL stores from platform-independent data models* [1] was presented. This approach describes a process based on models that need as input the conceptual data model and the queries that will impact the final database. Subsequently, the derivation is made to two logical models according to the type of database to be used (column family and documentary). Finally, a set of algorithms are applied, according to the logical model generated, to create the physical model corresponding to the NoSQL database engine to be used.

Secondly, *NoAM: (NoSQL Abstract Model): Data Modeling in the NoSQL World* [2] was presented. This approach proposes a phased design process. In the first phase the conceptual data model is made, and a set of aggregates are identified, in the second phase a logical model is proposed and finally, in the third phase the physical model is generated according to the NoSQL database engine to be used.

Thirdly, the approach *Modeling NoSQL databases: From Conceptual to Logical Level Design* [3, 4] was presented. In this approach, a conceptual model was presented, which is generated based on a set of constructs and/or layers. Then, through a set of templates, the corresponding transformations are proposed to obtain a logical and physical model according to the NoSQL database engine to be used.

Finally, a comparative analysis is proposed where it is suggested which design process or processes are more convenient according to the type of NoSQL database to be used.

As future work, we intend to apply the design processes analyzed to different case studies and different types of NoSQL databases.

References

1. Alfonso de la Vega, DiegoGarcía,Saiz Carlos Blanco,Marta Zorrilla, Pablo Sánchez. Mortadelo: Automatic generation of NoSQL stores from platform-independent data models.Future Generation Computer Systems. Volume 105, April 2020, Pages 455-474.
2. Paolo Atzeni, Francesca Bugiotti, Luca Cabibbo, Riccardo Torlone. Data Modeling in the NoSQL World. HAL open science. https://hal.archives-ouvertes.fr/hal-01611628.
3. Shreya Banerjee, Anirban Sarkar. Modeling NoSQL Databases: From Conceptual to Logical Level Design. 3rd International Conference on Applications and Innovations in Mobile Computing (AIMOC – 2016) At: Kolkata, India.
4. Shreya Banerjee, Anirban Sarkar. Logical level design of NoSQL databases.2016 IEEE Region 10 Conference (TENCON).
5. Luciano Marrero, Verena Olsowy, Fernando Tesone, Pablo Thomas, Leandro Corbalán, Juan Fernández Sosa, Patricia Pesado: Un Estudio de Procesos de Diseño de Bases de Datos NoSQL. XXVIII Congreso Argentino de Ciencias de la Computación - CACIC 2022. ISBN: 978-987-1364-31-2. http://sedici.unlp.edu.ar/handle/10915/149452.

6. Pesado P., Thomas P., Delía L., Marrero L., Olsowy V., Tesone F.: Análisis de performance en Bases de Datos NoSQL y Bases de Datos Relacionales. XXVI Congreso Argentino de Ciencias de la Computación (CACIC 2020). ISBN 978-987-4417-90-9. http://sedici.unlp.edu.ar/handle/10915/114202.
7. Pesado P., Thomas P., Delía L., Marrero L., Olsowy V., Tesone F., Fernandez S. J.: Un estudio comparativo de bases de datos relacionales y bases de datos NoSQL. XXV Congreso Argentino de Ciencias de la Computación (CACIC 2019). ISBN 978-987-688-377-1. http://sedici.unlp.edu.ar/handle/10915/91403.
8. Migani Silvina, Vera Cristina, Lund María Inés. NoSQL: modelos de datos y sistemas de gestión de bases de datos. XX Workshop de Investigadores en Ciencias de la Computación (WICC 2018, Universidad Nacional del Nordeste). http://sedici.unlp.edu.ar/handle/10915/67258.
9. MongoDB. https://www.mongodb.com/es. Abril de 2023.
10. CouchDB. https://couchdb.apache.org/. Abril de 2023.
11. Neo4j. https://neo4j.com/. Abril de 2023.
12. OrientDB. https://orientdb.org/. Abril de 2023.
13. Redis. https://redis.io/. Abril de 2023.
14. Amazon DynamoDB. https://aws.amazon.com/es/dynamodb/. Abril de 2023.
15. Apache Cassandra. https://cassandra.apache.org/_/index.html. Abril de 2023.
16. Apache HBase. https://hbase.apache.org/. Abril de 2023.