Facultat d'Informàtica de Barcelona

Discovering Ship Navigation Patterns towards Environmental Impact Modeling

Alberto Gutiérrez Torre

Supervisors:
Josep Lluís Berral García (BSC)
David Carrera Pérez (BSC-UPC)

Barcelona Supercomputing Center (BSC)
Data Centric Computing group

Master in Innovationand Research in Informatics
Data Mining and Business Intelligence specilization

April 28, 2017

Abstract

Ship positioning and maneuvering information is highly relevant to understand the levels of pollution on coastal cities and sea-life quality, containing latent patterns of vessels behavior, that are of utility on earth sciences and environmental research.

Using Automatic Identification System (AIS) data enables air quality models to have finer grain estimations. However, the data as it is, carries uncertainty and errors. Therefore, there is a need for a methodology to filter and clean it and to extract patterns. Ship navigation traces can be understood as time series. Here, we present a methodology for characterizing ships by their navigation traces, using Conditional Restricted Boltzmann Machines (CRBMs) plus classic clustering techniques like *k-Means*.

From the inputs received from ships using the AIS, containing ship positions, speed, and characteristics, we produce a processed cruising trace that a CRBM can encode while preserving the time factor and reducing dimensionality of data. Such codification can be then clustered or pattern-mined, then used not only for ship classification but also to cross such behavior patterns with environmental information. In this paper we detail such methodology and validate it using data from the Spanish Ports Authority records from national and international fishing vessels and passenger and cargo ships.

Along the pattern mining methodology we propose how to use Apache Spark for the data cleaning process until it arrives to the Conditional Restricted Boltzmann Machine (CRBM). Finally, we develop a visualization tool for data exploration and pattern evaluation.

Contents

Chapter 1

Introduction

In a recent study by Mueller et al. [1] performed in the city of Barcelona, it is seen that *Air Quality* is a cause, amongst other factors, of premature death. The conclusions of this study have reached the citizenship through several newspapers, making them more aware of the direct effects of pollution on their life.

Figure 1.1: Pollution cloud over Barcelona. Photography taken by the Fabra Observatory's meteorologist, Alfons Puertas.

According to the European Community Shipowners Associations (ECSA) in 2015, maritime traffic has become a key component for European economy [2], being sea transportation more fuel-efficient than other modes of transport (e.g. trucks and trains). Also, according to a recent report presented by the International Maritime Organization (IMO), it is expected that this form of transport will continue increasing in the future due to globalization and the increase of global-scale trade [3] but at the same time, it is considered an im-

portant contributor to primary atmospheric emissions in coastal areas [4] and subsequently to European coastal air quality degradation [5], especially in the North Sea and the Mediterranean basin. Maritime traffic is also responsible for about 2.5% of global greenhouse gas (GHG) emissions, which are predicted to increase between 50% and 250% by 2050 [3].

The CALIOPE[1] air quality forecasting system is a state-of-the-art modeling framework that integrates a meteorological model, an emission model, and a chemical transport model to simulate air quality concentration with a high spatial (up to $1km^2$) and temporal (1 hour) resolution. The air quality results are continuously evaluated with a near real time system based on measurements from the Spanish air quality network, and the performance of the system has been previously tested in different evaluation and air quality management studies [6]. The HERMES model is the emission core of the CALIOPE system and has been fully developed by the Earth Science department of the BSC [7]. Due to the high impact of maritime traffic on ambient pollutant levels at the urban area of Barcelona [8], one of the current objectives of the group is to improve the emission estimation of this activity using an AIS-based methodology.

AIS is the Global Positioning System (GPS) based tracking system used for collision avoidance in maritime transport, as a supplement to marine radars. AIS provides information such as a unique identification for each ship (Maritime Mobile Service Identity (MMSI) and IMO number), the position as latitude and longitude (GPS positioning) and speed (from the on-board gyrocompass). Such information is used by maritime authorities to track and monitor vessel movements, from AIS base stations located along the coast, and transmitted through standardized VHF transceivers. According to the IMO's Convention for Safety of Life at Sea, AIS equipment is required to be installed in all international voyaging ships with gross tonnage greater than 300 tons and all passenger ships [9].

Tracking ships through the AIS provided data not only can improve ship management and logistics in ports and maritime routes but also help to correlate ship presence and their *operation mode* against atmospheric emissions and sea life quality. However dealing with that data, given the amount of on-route ships and the high frequency of AIS updates (up to one every six seconds in most situations), requires employing *Big Data* techniques, understanding *Big Data* as those situations where big volumes of input overwhelm our commonly used methodologies, making us to change them for techniques designed towards automation, scalability, or approximation.

The field of Data Mining applies consolidated Machine Learning and statistical techniques for analyzing such data, extracting relevant values, frequent and rare patterns, and also model behaviors. Most of those wanted patterns are not trivial or present themselves at simple sight. They can even be found across huge amounts of data, unable to be handled exclusively by human experts. Discovering which patterns ships perform according to their position, speed, trajectory, and also given their properties, will allow to provide explanation to: 1) air pollutant concentrations in coastal zones and cities, and 2) degradation

[1] http://www.bsc.es/caliope/es

of sea life, by detecting unusual or even criminal activities from fishing fleets working in special sea-life protection zones. All of this knowing that AIS data is not always accurate as devices providing such information can fail, can be tampered or attributes that have to be manually inputed are wrong. This work tries to overcome this problems applying learning techniques over data.

1.1 Motivation

In this work we present a procedure for mining patterns from AIS data, looking for common behaviors depending on the vessel position, speed and ship properties, while having into account bathymetry information (sea floor depth) and coastal infrastructures (ports and docks). The main goal of this study is to characterize ships and other kind of vessels to detect common behaviors providing explanation to earth scientists for observed air pollution patterns on coastal cities and the degradation of the sea floor life due to illegal trawling practices.

Recent studies like Jalkanen et al. [10–12] show that AIS data can be used for the estimation of high spatial and temporal resolution maritime emissions. Moreover, AIS-assisted emission estimations can also be effectively used to assist policy design and corrective measures of a specific shipping sector (e.g. cruises and ferries) [13] and to improve the efficiency of ships [14]. Compared to traditional emission estimation methodologies, the use of AIS data provides information of instantaneous speed, position, and navigation status of vessels and subsequently allows for more accurate estimations of vessels' activities and the improved reliability of emissions and fuel consumption estimations [15]. Our current goal is to identify those common behaviors on ships depending on their position (docked, maneuvering in port, sailing in coastal regions or in open seas) and other attributes that allow us to classify their status for next studies on the contribution of such vessels to urban air pollution.

Further, one of the most sea-life endangering activities in the Spanish coasts (also in other coasts around the world) is illegal trawling. Trawling fishing ships drag large, cone-shaped nets through the sea at deep levels in order to catch fish. Such practice becomes risky in low depth stripes, as such nets may contact the sea floor also dragging corals and sea vegetation, causing severe damage to the ecosystem. Another desired goal is to find ways to detect such patterns, enforcing sea-life protection laws by detecting which fishing fleets are working with proper practices or bad practices.

The current work shows the proposed methodology applied in different scenarios, where for each one, patterns for ship status are identified, like fishing vessel discrimination, determining and correcting maritime status, and finding common patterns in specific geographic zones. Seeing the opportunities from the obtained models and patterns, we expect to improve modeling on pollution in future studies.

Despite the advantages offered by this approach, there are also some shortcomings associated with the AIS data that have been highlighted in a different work [16, 17], including data gaps, anomalies, and lack, in general, of data qual-

ity. In certain occasions not all of the data fields are fully or correctly populated (e.g. navigation status is not reported or it is incorrectly reported, speed calculated from real AIS information reaches unrealistic values). This fact may affect the suitability of AIS data for estimating air emissions around ports. In order to correct and refine the AIS information, data analytic tools and machine learning techniques are presented.

1.2 State of the art

As previously indicated, studies like J.P.Jalkanen et al. [10–12,32] presented the relation of ship traffic and exhaust emissions in the Baltic Sea, using the AIS mechanisms, also relating them to ships activity changes.

Pattern mining for GPS traces is a common practice in very different fields, looking for specific patterns in movement and behavior. Works like W.Qiu et al. [33] describe a methodology for mining patterns through Hidden Markov Models, producing semantical information to feed frequent pattern mining methods. Such work is also based on discovery of frequent episodes in time series [34], with the goal of discovering patterns series of events. Use cases for such techniques are social mining and recommendation [35], animal movement patterns [36], or elder care [37].

There are several studies about analyzing the semantics of GPS traces as in the work of Parent et al. [24]. In the case of this study, the only points of interest available at the moment are the 3 ports that are available in our data and the ships do not tend to go from one to the other. Therefore this approach was discarded. This approach will be considered for further analysis as we may want to define points of interest like common fishing spots or areas where the ship is waiting for a given service, e.g. for a tug to come and guide it to the allocated area in the port.

Here, we proceeded to find common patterns using CRBMs as a base for time windows, feeding them from GPS and other input sources, for discovering discriminating behaviors on a geographical space. The Conditional Restricted Boltzmann Machines (CRBMs), as probabilistic models derived from Restricted Boltzmann Machines (RBMs) [38] [39], are used in a wide range of problems like classification, collaborative filtering or modeling of motion capture, developed by the team of professor Geoffrey E. Hinton at the University of Toronto [21] [40] [41] [27]. Such models are usually applied for problems where time becomes a condition on data, i.e. time series. Other works like X.Li et al. [42] and Lee et al. [43] use the models for multi-label learning and classification. Based on their experiences and techniques, we are taking advantage on CRBMs, considering RBMs conditioned to the memory of our input data, this is, applying to our time-series a limited time-window.

1.3 Contribution

Summarizing the current contribution, here, we provide a methodology for applying CRBMs to encode series derived from AIS traces having into account time and position, to train clustering models for behavior discovery and classification. We show how such behavior classification can be used for ship profiling and characterization and how different use cases can benefit from this methodology by discovering useful patterns. Moreover, we propose a methodology for doing this kind of process using a Big Data architecture and also how to validate the found patterns using a visualization tool.

This work is structured as follows: Section 2 gives the background of the problem and proposed solutions for each part as well as AIS mechanisms and usages. Section 3 introduces the fundamentals of CRBMs and K-Means, along with a brief introduction to *Apache Spark*. Section 4 describes further details on each part of the problem and gives details on the implementation done including details on parallelism. Section 5 details the experimentation and validation of our methodology and use case scenarios. Finally, Section 6 summarizes this work, presenting the conclusions and future lines of work.

Chapter 2

Problem statement and proposed approaches

2.1 Background of the problem

Our aim is to leverage environmental modeling using an architecture that can handle AIS data, machine learning and statistical algorithms in order to obtain useful patterns that can be applied to improve the models and be used for other related use cases. As this problem is *data-centric*, the first step is to define the dataset.

2.1.1 Dataset definition

The currently used dataset has been provided by the Spanish Ports Authority (*Puertos del Estado*), from their vessel monitoring database collecting the AIS signals from all registered ships navigating national waters. Such database collects the information periodically sent from all registered vessels, and can be used by local port authorities in a network of AIS data-sharing. The dataset used for current experiments is a slice of data concerning the coastal area of Barcelona, including a week of maritime traffic. It is composed by more than 1.5 million entries and indicating 19 features, including the vessel identification, the position in longitude and latitude degrees, speed over ground, facing position, and other vessel properties like vessel category.

In Spain, *Puertos del Estado* has deployed a network of AIS base stations through the whole Spanish coast [18], with the dual objective of obtaining maritime traffic information (especially at the port area) and applying the AIS capabilities to navigation aid[1]. Each AIS base station is responsible for receiving the AIS messages from Very High Frequency (VHF) radio signals within its coverage area and sending it to the central hub for processing, storage and subsequent distribution to other AIS networks or interested users.

[1] http://redais2.puertos.es

The dataset provides three different ID variables to identify each vessel:

1. Name of the ship as given by the owners.
2. IMO number, unique ID provided and regulated by the IMO.
3. MMSI, unique ID provided and regulated by local agencies, e.g. BoatUS (Boat Owners Association of the United States).

There are two AIS device classes (A and B) differing in transmission power and capabilities, being Class B smaller and more short ranged than Class A. Ships transmitting with a Class B device are not required to have an IMO number, thus having *Not Available* values (NAs) in our data, we should focus on using the MMSI as candidate key identifier. However IMO number is available in some ships which do not have MMSI. Hence we need a mix of the two attributes to identify all the ships.

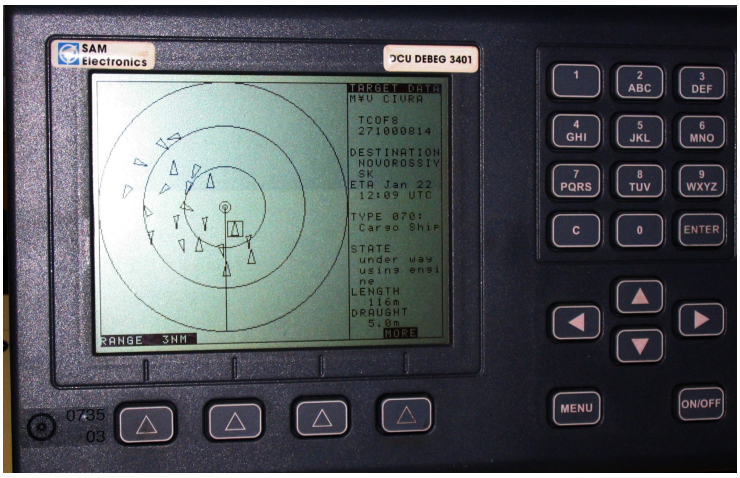


Figure 2.1: Example of AIS data visualization with on-board DCU used for collision avoidance. Photography taken by Clipper, extracted from Wikipedia's AIS page.

Moreover, AIS devices are always transmitting their properties like size in six different variables (length, beam, draught...) periodically so authorities and other ships know the vessel properties at each time. This data is considered as *static* and it is not of interest for the current work but may be considered in future works.

The *dynamic* status information provided by the AIS transmission has the following attributes:

1. Time-stamp: Time in which each sample of data was transmitted.
2. GPS Coordinates (latitude, longitude): The pair of coordinates in Latitude - Longitude degrees.

3. Speed Over Ground (SOG): Speed of the boat, measured as effective over ground, by having into account the tidal drifting or speeding up/down the ship, measured in knots with a resolution of 1/10 knots and a range from 0 up to 102 knots.

4. Course Over Ground (COG): Number of degrees rotated, measured as effective over ground having into account the tidal drifting, relative to the true north with a resolution of 1/10 degrees.

5. Rotation speed (Rot): Number of degrees rotated per minute.

6. Heading: The direction and sense traced by the ship respect to the gyro-compass, always pointing north.

7. Navigation Status (Navstatus) : A standardized identification of the current status of the ship. This feature is manually set by the crew. This denotes the susceptibility of such feature to errors and missing values.

8. Type of ship and cargo (Type): A combination of two integer values, encoding the type of ship and materials that it is currently transporting.

9. Draught: maximum number of meters of the actual draught in 1/100 meter scale.

Having time-stamps available, we can consider the ordered collection of transmissions from a single vessel as a time-series, and studying its evolution along time. Table 2.1 shows some sample data included in our dataset.

2.2 Challenges and proposed approaches

The requirements that this work tries to fulfill can be summarized as the following objectives:

- Apply a methodology for finding useful patterns for improving the quality of emission models and for improving our knowledge on other use cases, e.g. illegal trawling.
- Create a cleansing pipeline in order to provide the best possible data for the previous task.
- Create a visualization tool that enables to better know the data and analyze the patterns found.
- Build an architecture that can handle large amounts of data and complex processing.

In the following subsections, these tasks are analyzed and a solution is proposed for each one.

ID	size_a	size_b	size_c	size_d	length	beam
1	62	126	13	15	188	28
2	17	19	7	1	36	8
3	4	16	4	2	20	6
4	5	16	2	4	21	6
5	62	126	13	15	188	28
6	10	15	5	2	25	7

ID	draught	sog	cog	rot	heading	navstatus
1	7	5.50	317.00	127	326	0
2	3	0.00	170.00	0	47	8
3	4	10.00	220.00	-128	511	7
4	0	9.40	136.00	0	135	7
5	7	5.50	317.00	127	326	0
6	0	10.90	134.00	0	133	15

ID	type	latitude	longitude	time-stamp
1	70	40.91	2.47	2014-04-13 23:59:32
2	37	41.53	2.44	2014-04-13 23:59:31
3	30	41.30	2.19	2014-04-13 23:59:33
4	30	41.05	1.26	2014-04-13 23:59:33
5	70	40.91	2.47	2014-04-13 23:59:35
6	30	41.03	1.27	2014-04-13 23:59:35

Table 2.1: Sampled data from the dataset. Identifiers are surrogates from the real MMSIs. The rest of IDs are removed to maintain privacy.

2.2.1 Challenge 1: Enhancing data with patterns

Positioning and maneuvering of ships of any size and characteristics can be understood as time-series, displaying patterns and characteristic that could provide further knowledge, e.g. common distinct patterns for each specific category of ship (i.e. liquid and dry bulk carriers, containers, general and RoRo cargos, cruise ships, ferries, fishing vessels, tugs). Finding frequent patterns in such behavior from positioning and maneuvering can result in not only classification of ships, an information usually available, but information to be correlated with environmental variables like air pollution, quality of sea-life, or even used for docks management.

As said before, AIS data can be considered a *time-series*, as each input updates the vessel status in time. There are several approaches for mining patterns for time series, from stream mining methods for learning on time-changing data [19], to series-aware neural network methods like *Recurrent Neural Networks* and *Hidden Markov Models* [20]. Here, we are focusing on a simplistic pipeline consisting in CRBMs to deal with time dimensions [21] and a classi-

cal clustering method like *k-Means* [22]. The reasons for choosing CRBMs is because our analytics goal passes to determine patterns through dimensionality reduction attempting to simplify clustering and pattern mining processes. CRBMs have the ability to encode multidimensional input data and its history into a dimension and time-aware k-length code, easier for feeding standard and consolidated clustering techniques.

We have also considered other two conceptual alternatives. The first one is based on a common approach to genetic sequence analysis and other related fields pattern matching. To apply this idea, we need to transform the traces into sequences of movements and to do so we create a grid with the map so all the traces of the ship can be defined as movements on that grid like in a chess board and define the movements using the cardinal points. With this procedure, what we would obtain would be a sequence of movements defined by the cardinal points, e.g. ESSSW, means that the ship moved 1 to the east, 3 to the south and then 1 to the west. After this, applying the PrefixSpan algorithm [23], we would obtain the most common patterns. This idea was dropped as it was not flexible enough for this task as, for example, there was no way to introduce auxiliary variables that provide more information.

On the other hand, there are several studies about analyzing the semantics of GPS traces, as in the work of Parent et al. [24], in which the main idea is to define points of interest to analyze how the elements that generate the traces, i.e. cars, ships, persons, go from one point to the other. In the case of this study, the only points of interest available at the moment are the 3 ports that are available in our data and the ships do not tend to go from one to the other. Therefore this approach was discarded. This approach will be considered for further analysis as we may want to define points of interest like common fishing spots or areas where the ship is waiting for a given service, e.g. for a tug to come and guide it to the allocated area in the port.

2.2.2 Challenge 2: Cleaning and normalizing Data

With respect to the quality of the data it, is known that it contains errors. There are two kinds of errors: device or machine errors and human errors. The former are caused by failures on the instrumentation available on-board, e.g. a failure of the gyroscope, and can be present in several attributes like in *rotation* or *speed over ground* as they are automatically provided to the system without human validation. On the other hand, there are attributes that are manually set by the crew like navigation status so these attributes are prone to error if there is no strong protocol set at the ship.

In Figure 2.2, there are two examples of the same variable failing: on the left, it can be observed that the rotation value is wrongly set, maybe due to calibration error, and on the right, we have an always constant value which is not possible as the ship rotates. In Figure 2.3, it can be observed at the left picture that sometimes values may contain outliers because of a punctual failure. Finally, in Figure 2.3, the picture at the right presents a wrongly set navigational status variable can be observed. In this case, the variable is set in

advance, as it can be seen that the ship is moored and anchored while moving which, by definition of those statuses, is impossible.

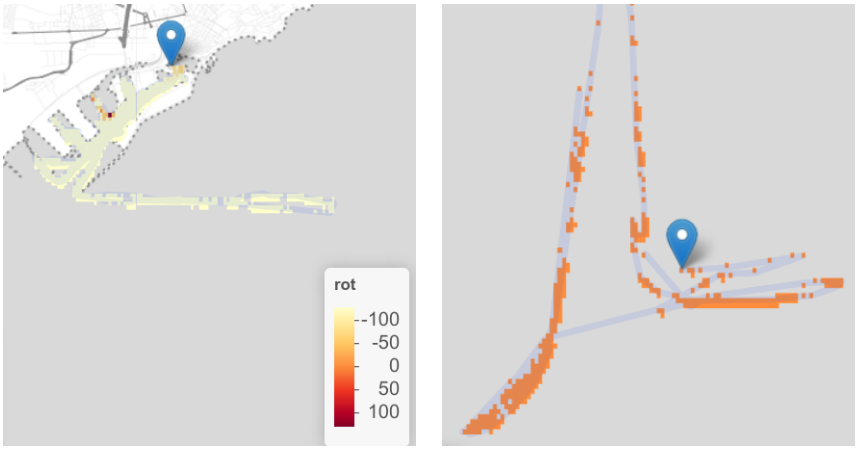


Figure 2.2: On the left: Rotation variable with value -128 in most of the cases. On the right: Rotation is always constant.

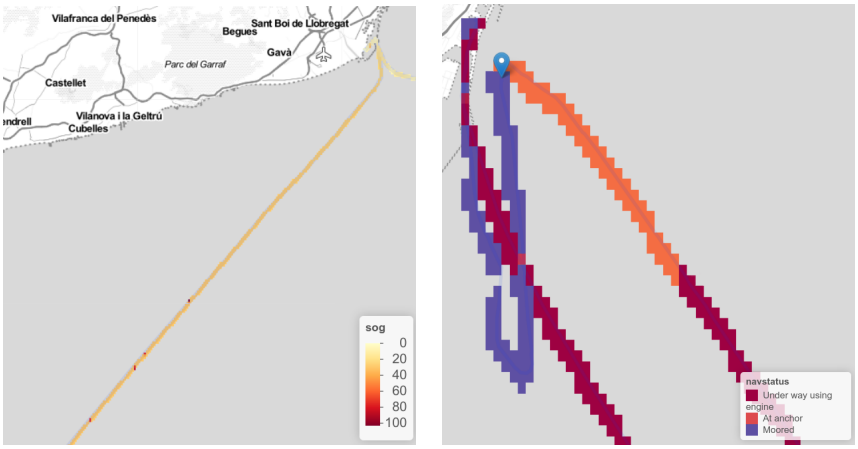


Figure 2.3: On the left: Outliers on Speed Over Ground variable. In this case, the usual measure cruising is around 20 knots, but punctually, it grows to 100, which is incorrect. On the right: Wrongly inputed navigation status. The ship is moving while moored and at anchor, which is impossible. Notice that it could be moved by the water currents when anchored but it would define an elliptical shape.

As some attributes are very hard to be fixed, we propose to extract some of these hard to fix features directly from the GPS traces, as we will see later in this section. The data cleaning procedure includes processes that can be executed per data entry and processes that require to have the whole time

series. After preprocessing, we have a time-series for each ship, with regularized times between observation and expanded features like relative positions and movement, allowing us to compare ships for their positioning and maneuvering, independent of the origin port or coastal point, even from length of some pattern repetitions. Also, absolute features are always available in case they are needed.

Sample level preprocess

The following preprocess steps are from the first kind:

- Date filtering and reformat
- ID generation
- Transforming navigational status into a readable string.

Dates may come in different formats, therefore we need to standardize them. When a sample arrives, if it is not properly timestamped and cannot be reconstructed, it is discarded because time is the most important feature when dealing with time-series as it provides the order of the events.

In most of the cases MMSI is present on the samples, however there are ships that do not provide this attribute. In this work we propose to use as identifier a new generated value that contains the MMSI whenever is available. If not it will contain the IMO number. If neither are available, an *NA* value will be set, so that the sample is not used as we can not properly identify its origin.

In order to provide a more readable output in the interface, the navigational status variable is transformed in its string representation, e.g. 0 is Under way using engine. Mind that this conversion is only for analysis of the results while building the application, as keeping the integer representation and transforming it in the visualization application is more efficient in the sense of storage.

Series level preprocess

On the other hand, the preprocess steps that require the whole series are the following:

- Split the series in sessions
- Resample time-series for time regularity
- Generate new features

In this data, it could be possible that a ship disappears for more than a given amount of time and then come back, e.g. ships going from Barcelona to Mallorca and then coming back. In this case, a problem arises when interpolating: if the ship is away for a long period of time, a line between the exit point and the new entering point will appear as if the ship would have been there. However this is false. In the work of Jalkanen et al. [10], a threshold is established: if the gap between samples is greater than 72 hours, the trace can be split in two in order

to prevent a huge amount of interpolated non-realistic data. In this work, this gap threshold is adopted by creating a new attribute called *session*. If the gap is greater than the threshold, the session ID from that point until the next gap will be the previous plus one. This way, we can split the trace and interpolate the data in a safer and more realistic way.

As seen before, there is a need to resample the time series as the methods we are going to apply requiere time regularity. In further sections we define how the process is done.

Next, in order to avoid bias or over-fitting on locality when searching for patterns, as it is shown in Section 5, a new feature is added indicating the relative movement, by obtaining the difference in Latitude - Longitude between each consecutive points. This way we register movements between registered observations instead of absolute values, having a movement feature free of geographical information. Also, the same procedure can be performed over the rotation feature, having as result relative rotation movements. However, rotation attribute from AIS is not always available or correctly inputed as shown in Figure 2.2, hence in this study we have created a rotation variable calculated from the GPS traces as they are more reliable.

Trawling is allow in depths between 50 and 1000 meters, as above 50 meters sea-life require special protection, and below 1000 meters it is unpractical [25]. Therefore, for this study an aggregated feature identifying the trawling zone has been added to the dataset and is used for the *illegal trawling use case*.

2.2.3 Challenge 3: Processing infrastructure

As this work is data-driven, one of the key parts of it is to define an infrastructure that processes data as fast as possible and that can handle large batches. The R language is a tool that is gaining momentum as *Data Science* gets more popular as it is quite powerful and easy to use once you learn how. However, vanilla R has a couple of flaws: all the process that is done only uses one thread of the CPU and is memory hungry. There are packages that solve the *mono-thread* issue, e.g. *parallel* or *snow* (both available in R CRAN repository), enabling R to execute tasks in several CPUs or even in different machines. This execution in several machines helps us dealing with the flaws commented before. However it is a very specific solution in which is hard to manage which resources is the application is able to use.

A framework that covers the functionalities required is *Apache Spark* as it provides a flexible parallel computation environment through easy to use Application Programming Interfaces (APIs), including an API for R since version 2.0 called *SparkR*. In Section 3.3 more information about this framework can be found.

2.2.4 Challenge 4: Visualization

Another requirement born from the need of better understanding the data is the visualization tool. GPS data is hardly understood from direct exploration

tools, e.g. R plots like histograms or box plots, that is why there are several tools that allow to plot GPS data over maps, as the region in which the point falls is of interest for the analysis.

There is a myriads of tools available to analyze GPS data, most under the name of Geographical Information System (GIS). However, we have not found a tool that is flexible enough that offers web exploration. This is the main reason to select R Shiny[2]. Shiny provides to R users an easy way to create web pages for interactive data analysis without knowing how to code web pages nor have deep knowledge of web elements. For map plotting, we have selected the *Leaflet* R package as it provides interactive maps on which the data can be plotted.

With this two packages, along with other packages for some functionalities, e.g. *Raster* package for creating rasters (grid images from data), we have developed a flexible tool that allows us to understand the data and validate the found patterns and that will allow the members of Earth Science department to easily navigate the data.

2.3 Summary

We have defined how the AIS data is and the requirements of the project. We have seen that there is a need of a data cleaning pipeline and learning techniques in order to achieve a good data quality for the modeling and we have proposed to use a cleaning pipeline of specific functions and the combination of CRBMs and k-Means for pattern mining and presented other alternatives. We have also seen that a Big Data architecture and a visualization tool are required and useful for performing the task.

In the next chapter, the required algorithmic and technical background is provided.

[2]https://shiny.rstudio.com/

Chapter 3

Technical background

3.1 Conditional Restricted Boltzmann Machines

Restricted Boltzmann Machines A RBM, in our case more concretely Gaussian Bernoulli RBM (GB-RBM), is the key building block of the CRBM. A GB-RBM is an undirected graphical model, as can be seen in Figure 3.1, with binary hidden units and visible Gaussian units, i.e. units that accept real values that may contain Gaussian noise as input, that models the joint log probability of a pair of visible and hidden nodes $(\boldsymbol{v}, \boldsymbol{h})$ as

$$\log P(\boldsymbol{v}, \boldsymbol{h}) = \sum_{i=1}^{n_v} \frac{(v_i - c_i)^2}{2\sigma_i^2} - \sum_{j=1}^{n_h} b_j h_j - \sum_{i=1}^{n_v} \sum_{j=1}^{n_h} \frac{v_i}{\sigma_i} h_j w_{ij} + C \tag{3.1}$$

where σ_i is the standard deviation of the Gaussian for visible unit i, c is the bias of the visible units , b is the bias of the hidden units, w_{ij} is the weight connecting visible unit i to hidden unit j and C is a constant. Notice that n_v and n_h refer to the dimension of $\boldsymbol{v}$ and $\boldsymbol{h}$ respectively. In practice we normalize the data to have zero mean and unit variance, we also fix σ_i to 1 as shown by Graham et al. [27] empirically when the data is normalized.

Conditional Restricted Boltzmann Machines The CRBM is a GB-RBM that models static frames of the time series modified with some extra connections used to model temporal dependencies. The CRBM keeps track of the previous n visible vectors in a $n \times n_v$ matrix which we call the history of the CRBM. The learned parameters of the CRBM are three matrices $\mathbf{W}, \mathbf{A}, \mathbf{D}$, as well as a two vectors of biases $\mathbf{c}$ and $\mathbf{b}$ for the visible and hidden units units respectively. $\mathbf{W} \in \mathbb{R}^{n_v \times n_h}$ models the connections between visible and hidden units. $\mathbf{A} \in \mathbb{R}^{(n_v \cdot n) \times n_v}$ is the mapping from the history to the visible units. $\mathbf{D} \in \mathbb{R}^{(n_v \cdot n) \times n_h}$ is the mapping from the history to the hidden units. Figure 3.2 shows a graphical representation of a CRBM.

Inference in the CRBM is performed using the contrastive divergence method. Further information can be found in G.Taylor's work [27].

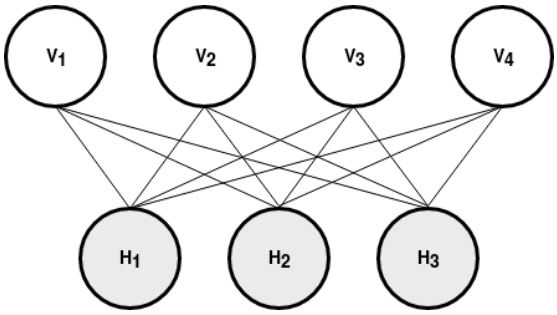


Figure 3.1: Diagram of a RBM. White nodes compose what is called the *visible layer*, which is the observed data. Grey nodes compose the *hidden layer*, which is the actual output.

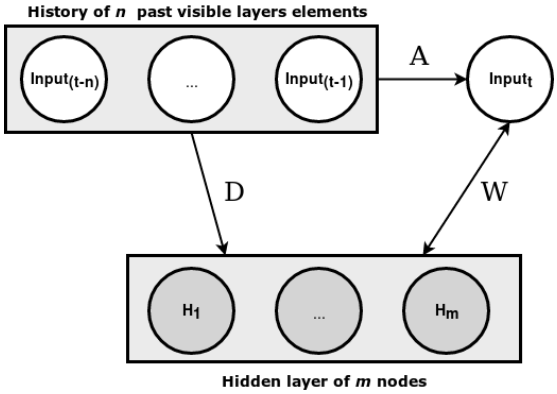


Figure 3.2: Diagram of a CRBM with history length of size n. Notice that in this case the white circles are not nodes but sets of nodes representing the visible layers.

3.2 k-Means

For the clustering process we have selected k-Means algorithm as is known to be fast and will provide the functionality we require. This algorithm is a prototype-based partitional algorithm which searches for k partitions that minimize the sum of distances between the elements and their assigned centroids. This algorithm is used to do clustering in n-dimensional space with continuous attributes as it is in this case. Notice that we are using euclidean distance in this case as first default approach.

The basic k-Means algorithm, also refereed to as Lloyd's algorithm, is as follows:

Algorithm 1 k-Means (Lloyd's algorithm)

1: Initialize k centroids given the criteria by the selected initialization function.
2: **while** Centroids change **do**
3: Assign the closest centroid for each point p of users
4: For each cluster recompute it's centroid
5: **end while**

Initialization function. As seen in Algorithm 1, an important step of K-Means is initialization. It is a heuristic algorithm so the quality of the partitions and the convergence speed depend on the initial solution given. However, as it is hard to define what a good starting solution is in general, amongst all the available methods there are two that are most commonly used: Forgy's function [28] and k-Means++ [29]. As Forgy's function is less restrictive and more widely used, this approach will be followed in the work.

Forgy intialization. This function, introduced by Charles Forgy [28], is a way to initializate by setting k random points of the dataset as centroids. It is defined as follows:

Algorithm 2 Forgy initialization function

1: Set every point from data into possible centroid list
2: **for** $i = 1 \rightarrow k$ **do**
3: Set a random point from possible centroid list as centroid i
4: Remove selected point from possible centroid list
5: **end for**
6: **return** K centroids

3.3 Apache Spark

With the growth of *Big Data* popularity, lambda architectures have grown in importance. This kind of architecture is built to balance latency, throughput and fault tolerance. It is generally applied when we have a cluster of machines, so processing can take place in parallel. This kind of architecture has three different layers:

- Batch layer: This layer gets the data batches and distributes them (if not already distributed) to get the processing done in parallel amongst the several worker nodes available. The result of the processing is stored wherever needed for further usage of the data. We use this layer when we have big amounts of data.
- Stream layer: This layer is similar to batch layer, however instead of dealing with big amounts of data, this layer deals with the data on the fly as it comes. The result can also be saved in a storage for further usage. We use this layer when we have data that comes from one or multiple sources in a streaming fashion.

- Serving layer: This layer is the one that the client applications will query in the end to get the data we have processed from both batch and stream layers.

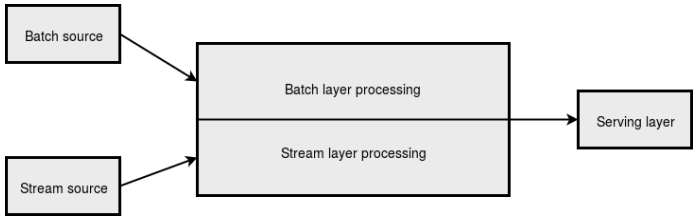


Figure 3.3: Lambda architecture diagram.

In our case we propose to use Apache Spark [26] as it is a framework that provides both batch and stream layers (even though the streaming layer is not pure streaming, it is in fact micro-batching, i.e. processing very small sets of data as they come, but not individual samples). Apache Spark provides APIs, for Java, Python, Scala and, since version 2.0, R, that enables the programmers to create *Big Data* enabled applications in a simple way without dealing directly with parallelism, i.e. programming with implicit parallelism.

As mentioned, Spark provides since version 2.0 the SparkR package, which allows us to use the framework using R language. This package adds a new variable type *SparkDataFrame* which represents a Spark *DataFrame* class, i.e. a distributed data matrix with named columns. With this type we can do several basic operations as if we were using vanilla R, but having the data distributed and processed in many nodes transparently. Moreover, it can work with User Defined Functions (UDFs) in two ways which work in the same sense as R *apply* function family.

By using this framework we an easy to use distributed computing and storage (using Hadoop Distributed File System (HDFS) in this case), which will be useful when we process future bigger datasets and it will be ready for having streaming data flows if the project jumps to Internet of Things (IoT) world.

3.4 Summary

In this sections we have seen how CRBM and k-Means works and a brief on the internals of Apache Spark.

In the next section we specify how is everything implemented.

Chapter 4

Implementation

In this chapter we define how the proposed solution is implemented. In this document we will only present the concept behind all the implementation, as the actual implementation can be found at the project's website[1] in a tab in the visualization tool. The main technologies applied in this part are the R language and its libraries and Apache Spark. All the implementation was done using *Jupyter Notebooks*[2] for better understanding and reproducibility of the whole process.

4.1 Data preprocess

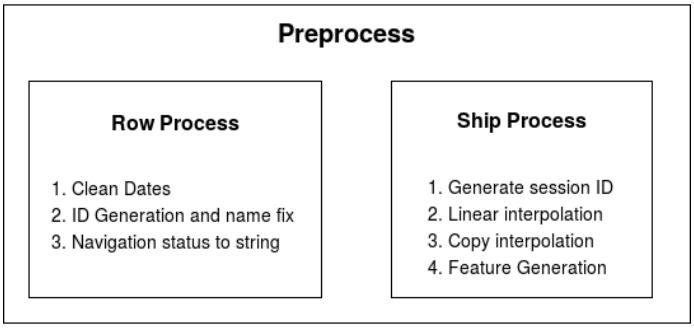


Figure 4.1: Diagram of the whole preprocess.

As mentioned before in Section 2, the preprocessing is divided in two blocks as shown in Figure 4.1. This conceptual division between processes that can be done for each sample and processes that require the whole ordered series for each ship makes us exploit parallelism as for the first one evenly distributed

[1] http://patrons.bsc.es:8080
[2] http://jupyter.org/

partitions can be used, boosting the process. Experiments on partitioning can be seen in Section 5.3.

4.1.1 Row processing

In this subsection we define the operations that are applied to the data for each sample without the concept of series.

Date filtering and reformat In this dataset there is no warranty to have correct timestamps as they are emitted from the AIS device but there is no check at reception. Initially, timestamps should come in YYYY-MM-DD HH:MM:SS format, e.g. 2016-01-01 00:00:02, but in some cases this time-stamp is wrong, e.g. instead of having a date, we have a floating point number which is not UNIX Epoch time-stamp nor the previously mentioned time-stamp.

The decision taken in this step is that when an anomalous time-stamp arrives we try to recover it using *POSIXct* R type conversion, which is one of the possible types for representing timestamps inside R. If the procedure fails, it means that the time-stamp is not recognized, hence this sample is dropped as samples without proper time-stamp are not usable in the context of time-series.

ID generation and Name fixing In order to identify the ships we need to have a unique ID for each one. MMSI is present in almost every ship, however there are some ships that has this attribute missing, in which we set IMO number as when MMSI is missing, this ID is available in this dataset.

When name is missing, we set the generated ID in order to identify the ship in the visual tool. We tried another approach, which was to obtain the actual names from a website, VesselFinder[3] searching by the available ID and transforming the data into a dataset using web scrapping. This approach proved to be useful, however it is not practical for a real world deployment as we are relying in a website that can deny us service as this way of working is not how it is intended to be.

Integer represented strings: Navigational status As Navigation Status is in numerical form, but being actually a code, it is turned into a classification label, each one with its corresponding meaning (e.g. *Under way using engine* for code 0) for the sake of human readability while developing. As we said before, this is not the optimal approach storage-wise as strings require more bytes than a single integer, however as we were developing this project there was a need of knowing at each stage the status of the ships. In the future works, this kind of attributes should be stored as integer and translated in the visual layer.

This kind of process in general is easy to do in R. What we did is to transform the numeric attribute into an R factor and then change the factor levels by using level(variable), which returns the levels of the variable and can be used to overwrite the tags. After changing the levels of the factor, it is possible to

[3] https://www.vesselfinder.com

transform the variable directly to character, which provides us with the final result.

It is important to notice that in SparkR the serialization of factors is not possible, which is a common pitfall. Therefore we always have to send the data as character or numeric for this cases.

4.1.2 Ship processing

In this subsection we define the operations that are applied to the data dividing it into series.

Sessioning Linear interpolation for missing data points is also performed in the study made by Jalkanen et al. [10], obtaining a finer grain position for ship traces. They considered that gaps greater than 72 hours not to be interpolated. In their case they filled the gaps using estimations from engine average details.

In this case, before interpolating, we create a new variable *session* that represents when a given threshold of gap length is exceeded, as explained in the previous chapter.

In R we can implement this using *zoo* package *rollapply* using a window of two elements for finding where the threshold is exceeded and then creating the new variable from the *breaks* found. For example, if we have threshold violations between elements 2 and 3 and 5 and 6 in a time-serie of 7 elements, we would obtain a vector of IDs as the following: 1,1,2,2,2,3,3.

Linear and copy interpolation Working with time-series usually implies having data regularized in time, as many techniques interpret samples as steady and regular, more than sparse, occasional or even redundant. When using CRBMs with time as conditioner, each position in the *delay* (the window of data history) is supposed to be given a set of weights towards the hidden layer, then data values slide through the window facing new weights based uniquely on their position in history. This way, each position in the history window discretizes time in equal segments, so sparse data needs to be densified, and missing data must be interpolated or predicted.

In order to work with CRBMs we need to regularize the collected AIS data, providing the same time lapse between observations. As observations are in the degree of seconds to minutes (rarely hours), time series are linearly interpolated and completed when needed to a granularity of seconds. Then, the resulting series can be sampled or aggregated to the desired granularity. For this study we chose to produce one sample per minute, after examining information provided in the collected dataset.

Next step is to retrieve each time series per ship, then processed for time regularization and interpolation when required. Data goes through a three step procedure:

- Expand phase: The series are completed by creating samples in the highest time granularity available (in this case, seconds) among the actual sam-

ples. E.g., two consecutive entries with six seconds of difference between timestamps will have five empty entries between them.

- Filling phase: The new entries values are populated through linear interpolation of the closest values from each side. E.g., in the previous case if the first actual value is 1 and the second is 7, the five empty values would be interpolated as 2,3,4,5 and 6 respectively.

- Reducing phase: Now the data is upscaled to reduce the size of the time series into a more handleable data, by sampling the series into periodical higher time ranges (e.g. minutes). In this current case of study a sample per minute is obtained by getting the first value per real minute, this is second 00.

For the linear interpolation we have relied in the *zoo* [31] package as it provides several functionalities for merging time-series and doing the linear interpolation.

As *zoo* interpolation does not take into account categorical variables, we need to create another process that samples the series in the same way but for this kind of variable. We implemented what we call *copy interpolation*, which is as simple as doing the same process but instead of doing the linear interpolation in the *filling* phase, we copy the last value available for each entry.

Feature generation In this work we are in need of some extra features generated from the original set. In the following list all the used variables and the process to create them are described:

- Relative Latitude-Longitude: This variable is extracted doing a subtraction of the latitude and longitude of a given sample by the same variables of the previous sample. The first value for relative latitude and longitude are set to 0.

- Rotation angle GPS: The angle of rotation is calculated as the angle (α) of two vectors, v and w, which are calculated as follows:

$$v = sample_i - sample_{i-1}$$

$$w = sample_{i-1} - sample_{i-2}$$

$$cos\alpha = \frac{v \cdot w}{|v| \cdot |w|}$$

$$\alpha = acos(\frac{v \cdot w}{|v| \cdot |w|})$$

- Accumulated rotation: We also tried to calculate the accumulated rotation using the previously calculated GPS rotation. This variable is calculated using a windows of N elements. For each sample, the accumulated value is the mean of the values of the current sample and the $N-1$ previous

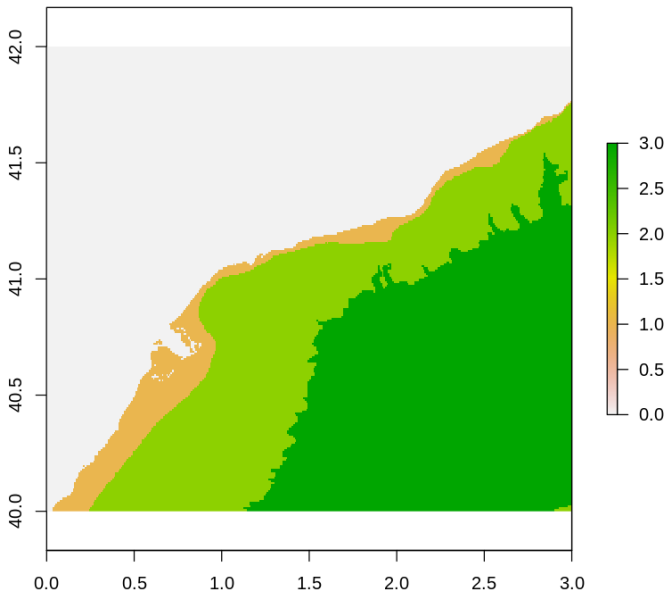


Figure 4.2: Example of coastal regions according to depth in the Catalan coast. The three regions separate coast from 50 meters, 50 to 1000 meters, and deep sea $>$ 1000m.

elements. This was calculated in order to check if the algorithm was resistant to punctual peaks, like is in the case of the rotation variable. However, in the end it was not needed as the algorithm performed correctly with the other variable.

- Bathymetry: This variable represents how deep is the water level in a given point. From a bathymetry raster map, i.e. a grid map with values on each cell, we have extracted the depth of the water for each point. In this case we discretized the depth of the sea according to the trawling fishing regulations [25]. In Figure 4.2 we can observe a raster which is the result of this discretization. There are three different regions, being the one in light green the one were is legal to do this kind of fishing. From this raster we extract the depth for each ship sample.

4.2 Pattern discovery methodology

The methodology here presented implements a data pipeline consisting in the preparation of data from Section 4.1, then passing the data through a CRBM for data encoding and reduction of dimensions, then clustering / classifying it through a classical method like *k-Means*. Following subsections explains each

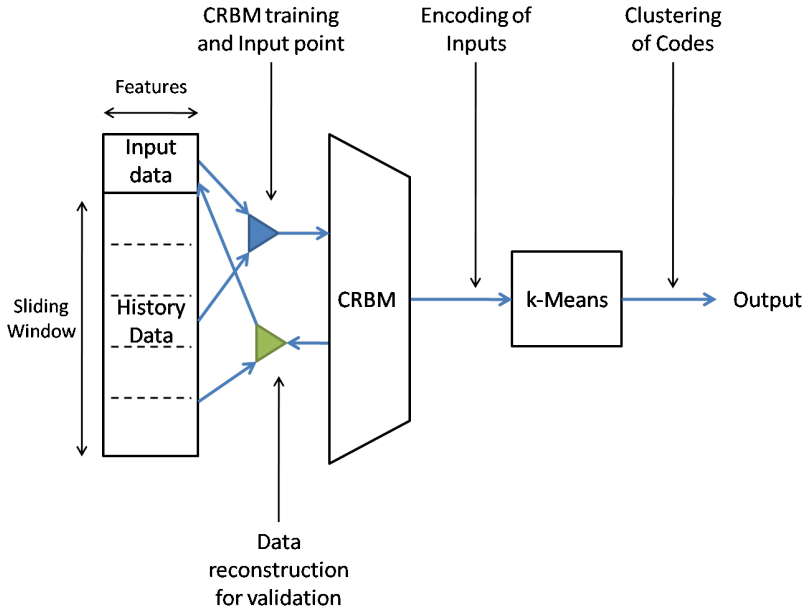


Figure 4.3: Schema of the data pipeline, given a sequence at a particular time t (and its n predecesor values) the hidden activations of the CRBM are computed. Then the hidden activations are fed to a *k-Means* in order to cluster the sequence at time t.

step of the chain, also depicted on Figure 4.3. The used CRBMs implementation can be found in Josep Lluís Berral's GitHub page[4].

4.2.1 CRBM Data Input

We understand the data from each ship as a time-series indicating, at each time period, the n_v attributes explaining the ship status. Let's consider a limited time window of size w, representing our memory or *history* of the CRBM. Timeseries in the data are not assumed to have the same length but the history length n must be less or equal to the length of the shortest time-series in the data. Given a multidimensional timeseries $input = (input_1, input_2, \dots)$, at time t we will have observation vector $input_t$ and $history_t$ will be $(input_{t-w}, \dots, input_{t-1})$. At time $t+1$, observation $input_t$ is pushed into $history$ while observation $input_{t-n}$ is popped out, therefore $history_{t+1}$ is $(input_{t-w+1}, \dots, input_t)$.

Notice that such mechanism implies "burning" the first n observations of each time series to have enough data for properly filling the *history* structure. We do not model the first n time steps of any time-series in the data.

[4] https://github.com/josepllberral/machine-learning-tools

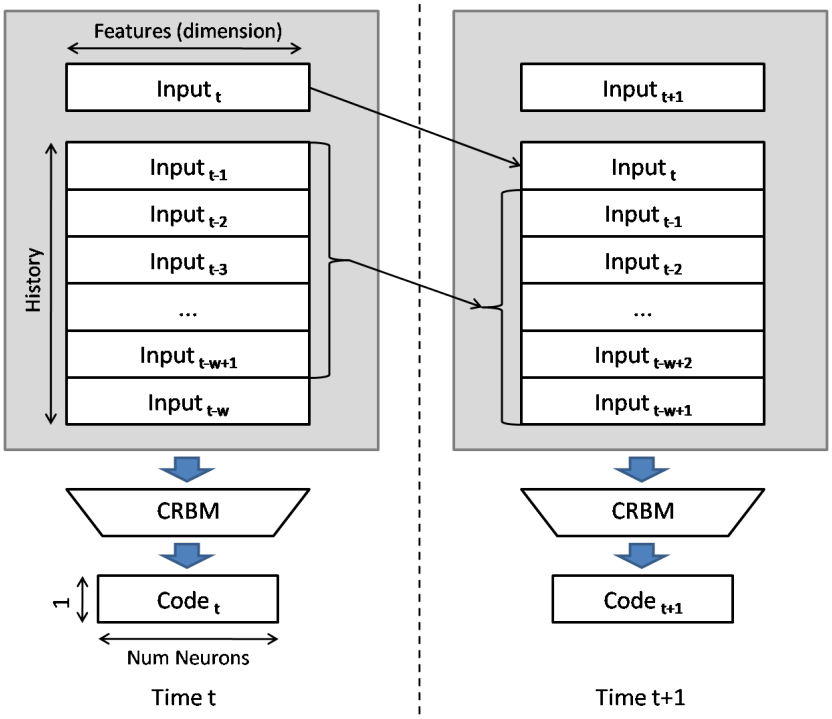


Figure 4.4: Schema of data sliding through the history.

4.2.2 Training the CRBM

The CRBM is trained with sample series of data, structured as explained in previous subsection. The CRBM is not aware of time by itself, but is our *history* input data what provides such notion. This allows training it through data batches and without forcing data order, once the notion of order is already present on each new instance. Best practices in modeling and prediction require to split training data with validation and testing data, to prevent the auto-verification of the model, so for this reason we performed this training process with a subset of the available time-series, to be shown in the experiments in next Section 5.

We will refer to the activations of the hidden layer for a given input as the output of the CRBM. Each instance passing through the CRBM is encoded into an activation vector of size n_h, being n_h the number of neurons in the hidden layer. This way, the ship tracking information and history are codified by a n_h-length vector, knowing that as far as a CRBM reconstruction misses the original data by little, such vector contains a compressed version of the current and historical status of such ship.

4.2.3 Training the clustering

Next step consists in finding common status vectors from different ships with similar behaviors. The principal hypothesis is that those ships performing similar patterns of position and maneuvering will produce similar codes. In other words, ships with similar behaviors in a given time window will be close in this n_h-space, and ships with very different behaviors will be far ones from others.

As a first approach, we decided to use a *k-Means* clustering technique, good enough for our purposes of finding clusters of data in n_h-dimensional spaces. The purpose of the clustering step is to categorize the status of each vessel at each time given their current status, having into account their recent status history, so for each input time-series we obtain an output time-series of status categories. Such time series characterize each ship, or parts of a ship's navigation. This let us to look for patterns in: 1) ships cruise and maneuvering, by looking for ships with similar sub-series; 2) geographical regions driving ships to behave in specific ways, e.g., ports where ships must behave in certain ways or fishing zones where ships slow down for trawling; 3) combinations of both, by detecting different maneuvers on specific fishing or protected zones.

In order to proceed according the good practices in modeling and prediction, training the *k-Means* has been done by separating training from validation and test data. In next Section 5 we will indicate the hyper-parameters for the applied clustering techniques given each use case.

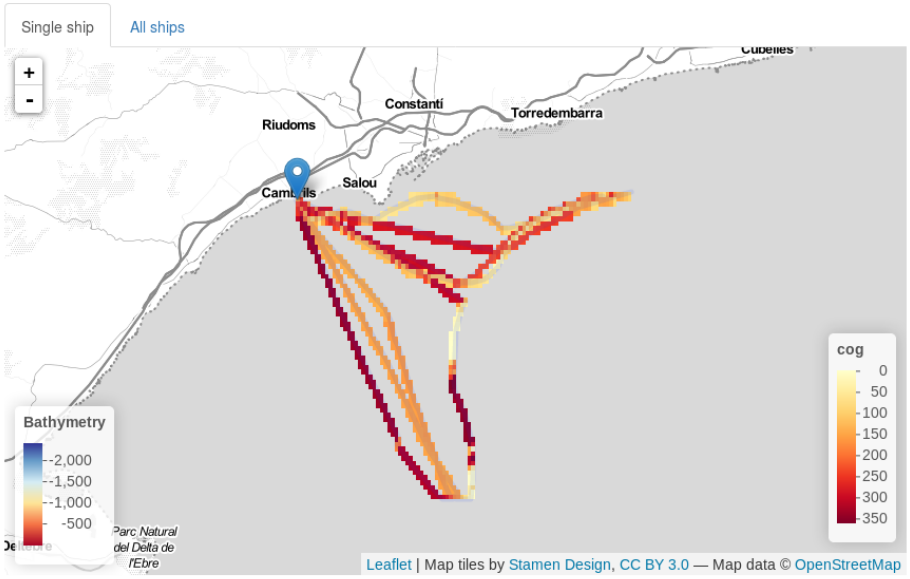
4.3 Visualization of results

Once CRBM and *k-Means* models are trained, new data can be fed to the pipeline, encoding and classifying each input into a status. At this time, similitude between patterns are done visually using a tool for ship trace visualization, created on behalf this and future analyses in our center. Such tool, available to the general public through a web service in the project's website [5], shows the geographical placement of each ship time-series along relevant variables like speed or orientation, also our output values like *cluster classification*. The visualization tool also provides the AIS traces used in the following experiments, courtesy of the Spanish ports Authority. Figure 4.5 shows an example of a time-series for a sampled vessel.

The visualization tool also allows the superposition of traces for different ships, allowing us to detect geographical regions where clustering labels *cluster*, indicating where behaviors are caused by geographical causes.

All the interface is implemented using R *Shiny* package. Shiny works as a reactive programming framework. Each visual element acts as an input or as an output. Input elements are what makes the interface interactive and gives the user a way to communicate values. On the other hand, output elements react to changes in input elements, re-executing the function assigned to it.

[5] http://patrons.bsc.es:8080/



Show 25 entries Search:

fechahora	mmsi	latitude	longitude	sog	cog	rot	heading	typeofshipandcargo
2014-04-14 04:39:00	224095660	41.05828	1.060433	9.700000	147.0000	-128	511	30
2014-04-14 04:40:00	224095660	41.05600	1.062415	9.800000	150.0000	-128	511	30

Figure 4.5: Example of the visualizing tool for traces and categorization.

The map is done combining Shiny with *leaflet* package, which provides a map on which elements can be plot. Variables are plot as rasterized images as plotting points directly with *leaflet* is not efficient.

As data input this visualization tool supports both SparkR (to read HDFS) and files from file system and it can handle more than one dataset for display.

4.4 Process parallelization

Parallelization in this kind of problems is crucial as large amounts of data take time to process. In this work we focus on the parallelization of the preprocess as it is a task that can be performed effectively using this approach.

As the functions are concentrated in two bigger functions, i.e. row preprocess and ship preprocess, we can use the apply family of functions over them. This kind of functions could be seen semantically as "apply this function over this set of data". If we manage to build our code using this concept, parallelization of the functions comes for free. The package *parallel* enables us to create abstract computation clusters which could be made of different threads in the

same machine or in other machines using an Message Passing Interface (MPI) interface.

After we construct the cluster object, the rest of work is to execute the previously mentioned apply but with some slight changes as shown here:

```
library(parallel)

nThreads <- 4

cl <- makeCluster(nThreads, type="FORK")
result <- parLapply(cl, data, funtionToBeApplied)

stopCluster(cl) #Free the resources
```

In our case whenever parallel is used we use cluster type *FORK* as it shares memory among processes, minimizing copies of data, however this type of cluster only is available when using a single machine, for several machines, as mentioned before, MPI cluster is required.

On the other hand, if we use this kind of approach in bigger clusters it gets hard to manage, therefore we have also implemented parallelization through *Apache Spark* using *SparkR* package.

```
library(SparkR)

## Open a session in SparkR 2.0
sparkR.session(spark.master = "yarn-client",
               sparkPackages = "com.databricks:spark-avro_2.11:3.0.0"
                );

data <- read.df(path, parameters...);
schema <- definitionOfTheSchema;

distributedResult <- dapply(data, functionToApply, schema)
result <- dapplyCollect(data, functionToApply)

distributedResult <- gapply(data, colsToPart,functionToApply,
    schema)
result <- gapplyCollect(data, colsToPart,functionToApply)
```

Parallelizing in *SparkR* works in the same way as in *parallel*, however there are three differences: Data is now distributed in our cluster (read.df reads a SparkDataFrame from, for example, HDFS), the result may or may not be distributed and the available apply functions are different.

The difference between *dapply* and *gapply* is that in the first the function will be applied for the whole partition that arrives to the node and, in the other, it is redistributed using the specified columns in *colsToPart* for the partitioning.

When using this kind of approach is important to take into account how the data is distributed amongst machines, as a bad partitioning of data can affect the performance, as can be seen in Section 5.3. SparkR offers a function, *repartition*, to recreate the partitions of data with a given criteria (number of partitions, partitions by id...).

Both apply functions benefit from this approach if the partitioning is done taking into account how both functions work and which is the process to be

done inside. In fact, if the partitioning is done by id, *dapply* works in the same way as *gapply*, however there is one difference: while *dapply* UDFs must only have one parameter, *gapply* functions have two parameters: key (value of the ID for that partition) and the data.

On the other hand, there is a difference between *normal apply functions* and *collect apply functions*. While *dapply/gapply* produce a *SparkDataFrame*, hence a distributed dataframe, *dapplyCollect/gapplyCollect* produce an actual R dataframe. This last kind of functions do the processing in parallel but in the last stage all the results are collected into the machine we are using to execute the code. In these functions the schema of the data is not requiered as R will interpret the results and Spark does not need to know about it as it will be treated as a serialized *black box* object.

In our case we use *dapply/gapply* as we want the result to be distributed. For example, if there is a need to get a single Comma Separated Values (CSV) we can always use the method *collect* in order to retrieve an R dataframe. Note that apply-collect functions can be very useful for debugging as there are functions in R that when applying them demote the type of the variables, i.e. transform complex types in basic types as could be when processing a *POSIXct* type which could be transformed into *numeric*.

We do not parallelize the training of the CRBMs as it can not be done using this approach, it should be done using Graphics Processor Unit (GPU) matrix multiplication optimizations, which is out of the scope of this project at the moment. However, the simulation of the CRBMs after training for obtaining the data that will go through k-Means is parallelized in a single machine using *parallel*.

4.5 Summary

In this section we have seen how the actual implementation of the clean pipeline is, including the division in row processing and ship processing, how is the combination of k-Means and CRBMs done and, finally, we have seen how is the preprocess implemented in Apache Spark and how we implemented the visualization tool.

In the next sections we present the experiments done for validating the project.

Chapter 5

Experiments

In this section the results of the project are validated. In Section 5.1 an exploration of the data is done for selecting which is the set of variables that is most appropriate for the task. In Section 5.2 the actual use cases are presented along the results obtained. Finally, in Section 5.3 performance of different computing approaches is evaluated.

In all the cases the experiments are performed using machines with two Intel Xeon CPU E5-2630 v4 @ 2.20GHz (20 real cores with 2 threads each) and 128 GB of RAM. Initial load of data is done from a Network File System (NFS) drive. This fact does not affect to the times shown in Section 5.3 as load time is not included in the metrics.

5.1 Feature evaluation and selection

The initial idea was to use plain latitude and longitude, as it carries implicitly speed because our time series are regularly sampled to the granularity of 1 minute, however this idea was discarded as this approach tended to produce more local patterns than general patterns.

In order to avoid locality we used the relative latitude and longitude variables, i.e. the movement vector produced since previous sample. In Figure 5.1 it can be seen that this variable loses the information about locality, however it still has the sense of direction.

This sense of direction produces an adverse effect as two equal behaviors are recognized as different as it can be seen in Figure 5.2. As the goal is to recognize patterns independently of the direction, another approach is needed.

We explored using *heading* and *rotation* for the model, but this two variables are very noisy and provide erroneous results as explained in Section 2.2.2.

In Figure 5.3 it can be seen that the linear interpolation approach is not valid for all attributes. In this case, as a big part of the series is lost, the rotation attribute is interpolated as if the ship would be rotating for a long period of time, however this is not the truth and it is far from it as it is impossible for

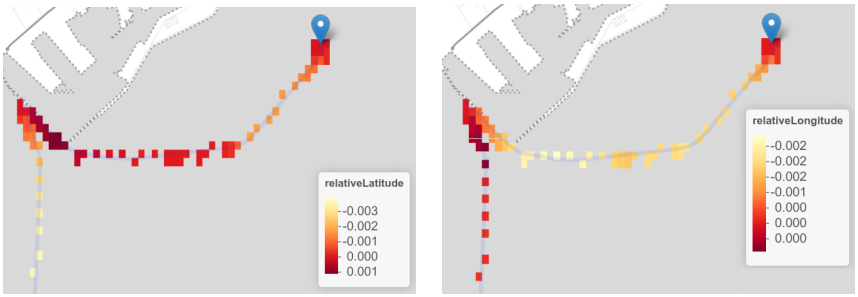


Figure 5.1: Relative latitude and longitude of a given ship.

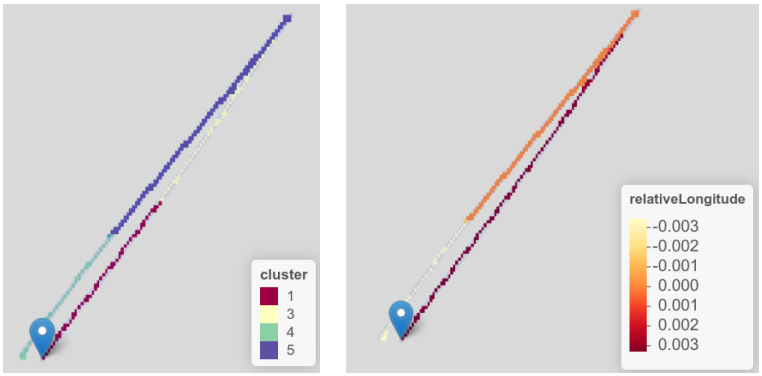


Figure 5.2: Patterns found using relative latitude and longitude. Direction is an important factor in this model.

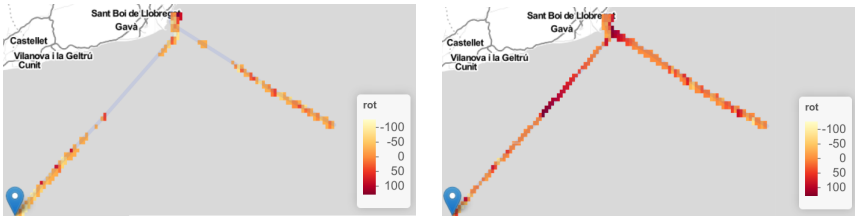


Figure 5.3: Original rotation variable versus imputed rotation variable

this kind of ship, i.e. a container ship, to do this kind of rotation. The ship photograph is present in Figure 5.4.

Moreover, in this case rotation attribute is also wrong as rotation is always 0, 127 or -127. If rotation is wanted in the model, we need to use the generated variable.

On the other hand, for *Speed Over Ground* attribute the interpolation works as the speed is almost constant during the trip as can be seen in left hand side of Figure 5.5. As we are studying big ships mostly, there are no fast changes of

Figure 5.4: Photo of Anna G. ship. Extracted from Marine Traffic website.

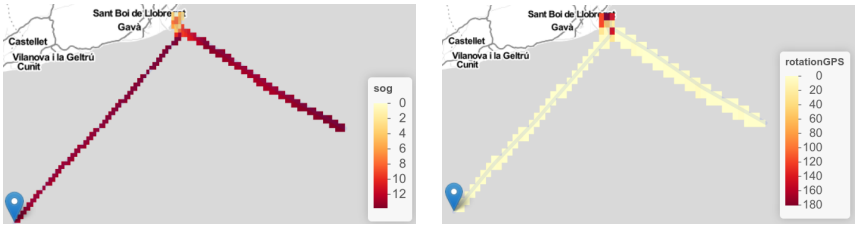


Figure 5.5: On the left: Ship's Speed Over Ground variable. Interpolation works. On the right hand side: Rotation variable for the same ship extracted from the GPS trace.

speed except when maneuvering, but rotation changes can be more spontaneous. If the speed of the ship decreases while it is not sending signal, e.g. because AIS system failed, this interpolation would not work either. As in this dataset we have not observed any pattern like this, as most of the interesting ships for the use case, i.e. cruises and fishers, do not fall in this category we will keep on using the linear interpolation, however in next steps of the project this part of the application needs to be improved.

Furthermore, we test the rotation attribute extracted from GPS traces. In the right hand side of Figure 5.5 we can observe that the rotation issues that had *rotation* variable are fixed in the new variable extracted from the GPS trace.

In order to add the concept of speed to the model, we propose to also use *Speed Over Ground* along with *GPS rotation* variable. Also, for the use case of trawling we will use the *bathymetry* variable.

5.2 Use cases

To illustrate the usefulness of the applied method and the current study, we focus now on experimenting with real use cases from the department of Earth Sciences at BSC, concerning some of the current problems found by the Spanish Ports Authority. Also targeting future studies about the impact of maritime emissions on air quality in the city of Barcelona, applicable to other coastal cities.

The experiments here show depict the following scenarios:

1. Fishing Discrimination: focusing in patterns for trawling ships and non-trawling fishing ships, also depending on the geographical localization.

2. Valid Vessel Status: determining the status of vessel directly from reliable GPS coordinates, to correct badly input *NavStatus* values.

3. Common Geographic Patterns: detecting characteristic patterns for specific geographical regions, for future air quality studies.

For such purposes, after several experiments with feature selection and refinement of aggregated features, we selected as input features the *bathymetry*, the *sog* (speed), and the *GPS-rotation* (the rotation calculated from the GPS positioning, as the feature *rot* is frequently missing or with incorrect values). Bathymetry is indicative of the geographical zone where vessels are navigating, if coastal zones, fishing zones, and open sea. Speed and rotation provide the movement vector of the vessel movements.

The CRBMs have been training by sampling a 0.66% of the ship series, and tested using the remaining 0.34%. To measure the CRBM errors (the capacity of encoding inputs with minimal error at data reconstruction), we use the testing series and perform a *simulation*, this is passing the series through the input + history sliding window, then through the CRBM for activation and reconstruction, and finally comparing the reconstruction with its input, using the Mean Squared Error (MSE).

During the experiments we attempted different CRBM hyper-parameters, with a wide range of hidden units in the hidden layer, and different *delay* or history window length. Using 10 hidden units in the CRBM, with delay around 20 observations (here minutes), provided us the best reconstruction results and differentiating clusters. CRBMs with higher hidden units provided low improvement or too much overfitting, and higher delay measures did not help clustering to identify fishing phases. The best configurations for CRBMs provided us an individual MSE for each dimension of $Err_{sog} = 0.0679$, $Err_{rotationGPS} = 1.4226$ and $Err_{bathymetry} = 0.7672$, being the best errors found among different tuning.

5.2.1 Use case 1: Fishing discrimination

This use case focuses on discriminating those fishing vessels performing trawling from those using other techniques. The goal is to automatically determine which

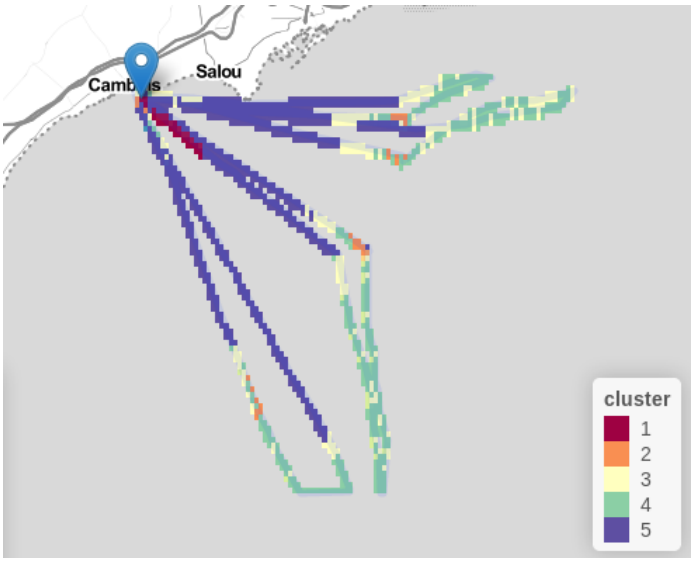


Figure 5.6: Example of status classification by clustering, on a trawling ship.

vessels are dedicated to each technique, and discriminate phases of sailing from phases of fishing. Superposing the bathymetry information (specifically the trawling zone) can result useful to discriminate such phases.

From the outputs of the CRBM, we can feed the clustering algorithm (here the *k-Means*). As hyper-parameter we selected a $k = 4$, as lower k provided differentiation between movement and resting, and higher k produced very similar clusters.

By applying the classification method over the testing ship-set, we can identify patterns for ships performing trawling not present in other fishing ships, cargos and passenger boats. A first cluster identified the docking and maneuvering movements, in port or turning around in high sea. A second cluster identified the cruising operations, moving at high speed in straight trajectories, and a third and fourth clusters identified slow movement with variations on rotation and trajectory.

Figure 5.6 shows and example of one of the trawling ships, where we can identify the previously mentioned clusters. Noting that Cluster 1 is an extra cluster for visualization, indicating the traces used for the CRBM history or delay, we observe the pattern of the 4 clusters inferred. Cluster 2 indicates the maneuvering in port and when shifting trajectories before and after trawling. Cluster 3 and 4 identify the movements during trawling, slower that regular sailing. Finally Cluster 5 is seen when speeding towards or from the fishing regions and the port

Such validation has been done by expert visual recognition of ship movement traces, and by identifying the vessels registry indicating whether they possessed trawling equipment on board. Now, using this labeling of ship status, we are able

to detect if a ship is performing trawling and at which moment. Other fishing vessels (seiners) show maneuvering phases on port and at sea when stopping for fishing (Cluster 2), also patterns of slow movement (Cluster 3) with consistent streaks, not like trawlers that tend to shift between clusters 4 and 3 when observing traces closely.

5.2.2 Use case 2: Determining a valid navigational status

The *NavStatus* feature, indicating the *navigation status*, is a value introduced by hand by the vessel crew. In regular cruisers or passenger boats, it is expected to be updates such *NavStatus* in a regular procedure, while fishing ships tend to be less accurate when registering such status. This use case proposes to focus on using the cluster labels as a parallel *NavStatus* indicator, to be compared to the real one and correct it when considered more reliable or when unavailable.

Table 5.1 shows the crossing of reported NavStatus, indicating those stopped due to anchoring and mooring, those stopped due to fishing, and those in movement. Such results allow us to validate the cluster labels: Cluster 1, as mentioned before, is the status for the data used as initial history, not classified; Cluster 2 refers principally to vessels mooring and in minor measure moving with their engines started, considering this maneuvering; Cluster 3 indicates those that are moving or fishing, and we visually detected that it is assigned to those moving towards fishing positions, or it is mixed with cluster 4 in trawlers; Cluster 4 refers to those moored or fishing, and we visually detected that such status is given to those trawling, moving much slower compared to other speeds (1/4 to 1/10 of regular moving speed); Cluster 5 is split between moving, fishing or moored, but by visualization we observed that those labeled as 5 are actually sailing towards fishing positions or returning to port.

	At anchor	Engaged in fishing	Moored	Not under command
1	0.03	0.18	0.09	0.00
2	0.07	0.17	0.50	0.00
3	0.02	0.26	0.11	0.00
4	0.11	0.22	0.47	0.01
5	0.06	0.24	0.32	0.00

	Restricted maneuv.	Undefined	Under way using engine
1	0.00	0.03	0.67
2	0.00	0.04	0.22
3	0.00	0.06	0.55
4	0.00	0.05	0.14
5	0.00	0.04	0.34

Table 5.1: Clusters vs. NavStatus labels. Values are normalized per row. Notice that Cluster 1 refers to the *delay* data not classified

We know that the NavStatus attribute is set *by hand*, making it susceptible of containing errors. Actually, fishing vessels usually set their NavStatus to

"engaged in fishing" at any time, even when sailing and mooring. Through this relations we can determine not only which clusters represent each real status, but we also can help to correct the real status of such ships. Also, for those without status ("undefined"), we can use the assigned cluster label as expected status, and applying approximate NavStatus labels by using the majority label for each cluster: indicating as "moored" if Cluster classifies it as 2, "under way using engine" if Cluster is 3 or 5, "moored OR slow fishing" if cluster is 4.

5.2.3 Use case 3: Common patterns by geographic zone

Finally, the last use case focuses on detecting geographical regions where a determined predicted status (cluster label) is predominant. Dividing the maritime space in quadrants (customizable in the visualization tool), we assign to each one the most present label. This way while some greater regions display mixed labels, others display uniformity of labels, identifying that label as characteristic for that region. Such characterized regions will become relevant in next studies for relating maritime traffic behavior with air quality models, as such observed pollution levels, modeled emission sources (e.g. industry, road transport) and meteorological parameters (e.g. wind speed and wind direction, temperature).

As we can observe in Figure 5.7, showing the collected traces of all ships labeled by our clustering, most of the traces concentrate around the 5 ports in range. Traces of Cluster 4 are totally visible in zones where trawling is permitted, displaying those ships fishing using that technique. Such ships tend to repeat their working pattern by departing from port, fishing, then returning to port, marked by the traces labeled by Cluster 3, concentrated around the port showing high traffic departing or arriving to port with low speed, with NavStatus "under way using engine". Cluster 5 indicates maritime paths for ships speeding to reach the fishing zones or directly returning to port. Finally, in the port areas inside the breakwater, Cluster 2 is predominant as expected, indicating ship maneuvering.

Is for the activity concentrated around ports (most of the points around refer to Cluster 3, also moving with engines on) that such region classification has great interest when modeling air quality.

5.3 Performance evaluation

5.3.1 SparkR and data distribution

Having SparkR does not provide directly high performance computing. Data locality is one of the factors that determine how fast our computation is going to be done. As there is few information about how data partitioning affects SparkR performance, we provide a small section of how partitioning affects the performance of the preprocess. We test three different cases: Preprocess without repartition, with evenly distributed partitions , i.e. with the same number of samples, and with the data distributed using the ship ID. SparkR executions are

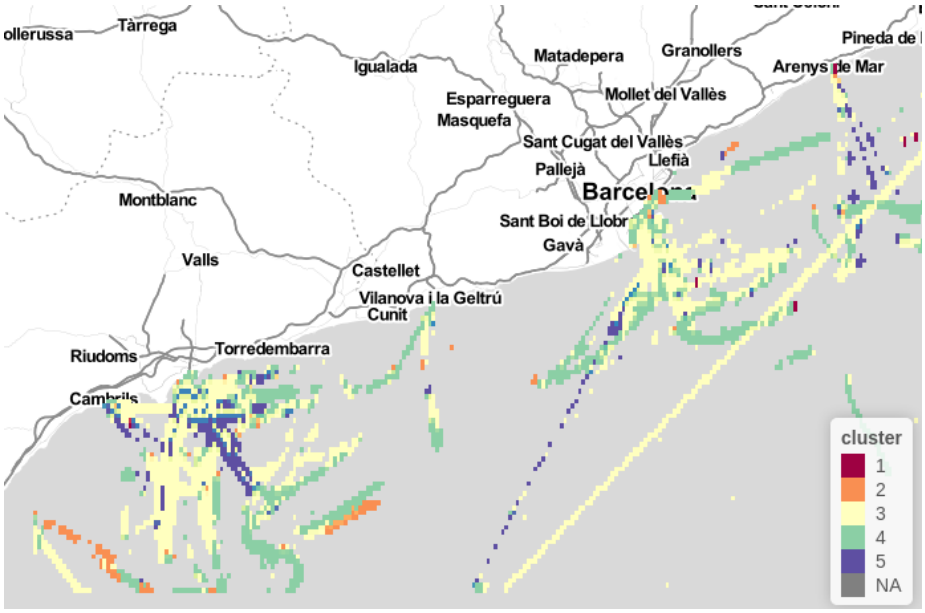


Figure 5.7: Map created with all the ships using the cluster variable. Cells with less than 10 samples are omitted for clarity.

done using 20 threads from 2 different machines in order to exploit the cluster component of Spark.

By loading directly the CSV directly without repartition we obtain a data frame with 30 partitions. For the evenly partitioned data frame we force the process to obtain 80 different partitions (number of threads times 2). In the case of the distribution by ship ID we obtain 413 partitions by using MMSI attribute as the generated ID is not available at that point.

In Figure 5.8 we can observe the distribution of samples amongst the partitions for each case. Except for the partition done by ID, the other are well equilibrated. By ID partition presents one partition with more than 120000 elements, this is because there is a set of ships that do not have MMSI and have only IMO number. This will affect performance as that batch will be processed by only one thread.

In Table 5.2 we can observe that using directly SparkR without considering the initial partitions is an error as the process is the slowest of the three in both cases. This is probably due to the fact that 30 partitions are produced and we have 40 threads in total, so 10 threads are actually not working. Even distribution seems the most reasonable partitioning by default in our case as it gives good results in both processes, however partition by ship is slower in the first part because of the partition with more than 120000 elements. In the case of the second part, partition by ship wins as it seems to reduce the data shuffling between nodes.

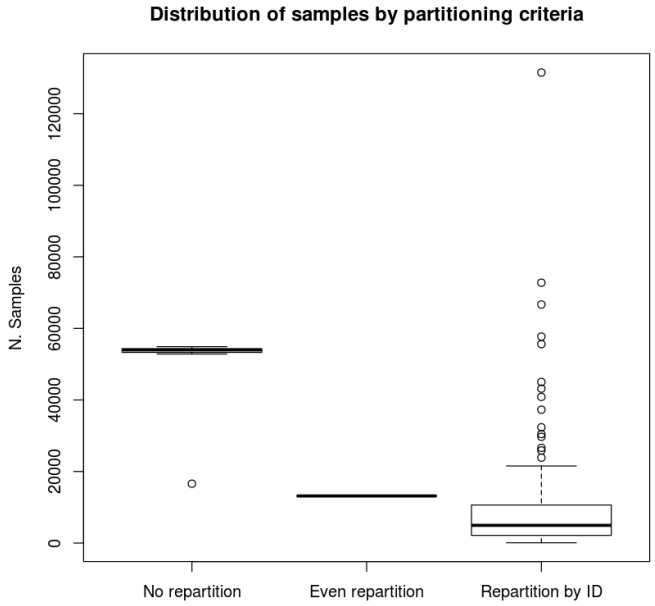


Figure 5.8: Distribution of the samples amongst the partitions using different partition criterions.

	No repartition	Even repartition	Repartition by ship
Row Preprocess	38.91 (0.34)	**8.46 (0.20)**	28.90 (0.29)
Ship Preprocess	131.61 (1.90)	116.54 (0.57)	**110.28 (0.35)**

Table 5.2: Time average and standard deviation (in parenthesis) for each part of the preprocess from 10 executions each.

5.3.2 Parallelization

Now we compare the performance of multithread approach with *parallel* package and SparkR. We also compare the results with the single-thread version as baseline.

As can be seen in Table 5.3, multi-thread and SparkR versions are about 12-17 times faster than single thread. Using 40 threads, a speedup between 30-40 times would be expected, however we have to take into account that the distribution of the data has to be done between processes and also that part of the data path (in the sense of the physical path in the machine's board) is shared amongst threads.

Between multi-thread and SparkR versions, in the row preprocess SparkR obtains the best result. This could be so as data is already distributed between

nodes. In the case of ship preprocess, multi-thread is faster. This could also be due to data distribution and overheads on communication of data and results.

	Single-thread	Multi-thread	SparkR (best time)
Row Preprocess	160.24 (4.67)	9.69 (0.50)	8.46 (0.20)
Ship Preprocess	1245.81 (17.63)	102.54 (3.68)	110.28 (0.35)

Table 5.3: Time average and standard deviation (in parenthesis) for each part of the preprocess from 10 executions each.

5.4 Summary

In this section we have presented an analysis on the variables and how they affect the pattern mining procedure. We have also applied the procedure to cover 3 different use cases and validated that it works. Finally we presented the evaluation of the performance of the different approaches, in which we can observe that *SparkR* is a good choice.

In the next section we present the overall conclusions.

Chapter 6

Conclusion and future work

6.1 Conclusions

Detecting and discovering patterns in maritime traffic is an important topic for modeling air quality in coastal urban zones and sea-life. Maritime emissions, combined with urban emissions (industry and road traffic), are responsible of pollution in such areas.

In this work we presented a methodology for characterizing maritime traffic, understanding it as time series, and using an ensemble of CRBMs and clustering techniques like *k-Means* to reduce dimensionality of data while considering time, then clustering it into common patterns of traffic. Such methodology implies pre-processing data, knowing that AIS provides error-prone data. Such datasets can be cleaned using standard techniques, also aggregated features can be derived from the most reliable ones, i.e. GPS traces.

CRBMs have proven to be useful for reducing such dimensionality, as most time series contained more than 3000 observations, even after pre-processing and reducing the time scale from seconds to minutes. When tuning the CRBM hyper-parameters, we observed that it is not required to introduce a large history window (< 20 minutes) or a high number of hidden units (~ 20) to achieve good results. Also *k-Means* appeared as a simple but effective approach to clusterize the reduced outputs of CRBMs, comparable to real ship statuses.

By using the presented methodology, we observed identifiable patterns for real use cases, like vessel discrimination and operation modes. Such patterns can be used to complement or correct missing or erroneous data from AIS, trace ship behaviors and recognize their activity, and define geographical regions with common operation modes and behaviors. We also provided a tool for data and patterns visualization, available to the general public.

Finally, a first step into a Big Data architecture is defined, testing it against another parallelism approach. We have seen that this kind of architecture works and if we want an scalable way of processing this type of data, this approach is useful.

This work has been submitted to the ACM SIGKDD International Conference (Knowledge Discovery and Data Mining), to the Applied Data Science track [1]. Also, an extended version is being prepared for a journal publication.

6.2 Future work

This approach has proven to be useful, however the data used in the experiments is only one week of data. In later stages of the project we will be able to get one year of data in order to validate the methodology and to extract actionable knowledge for the earth scientists. After finding patterns and validating the used methodology it would be a good idea to try to extract a *grammar of the movement* of the ships, taking the patterns as symbols and their succession as relationships between them, in order to extract further knowledge from the data in a simpler representation.

In the field of feature extraction and cleaning there is still more work to do. In next steps we will study and apply smoothing techniques like Kalman Filters in order to remove punctual failures, i.e. anomalous peaks on the data, thus improving the performance. There is also a need on improving interpolation in attributes like *Speed Over Ground*, in which we could use the Haversine distance and then calculate the speed for the missing gap. Also, we would like to go further from linear interpolation and test other methods like Akima interpolation, based on splines, in order to obtain smoother interpolations.

On the side of architecture there is still work to do. About storage, we need to study the available databases that can work with Spark and design the schemas and interaction with the code in order to obtain better performance by having control on where is the data stored and a standard gateway to access the data from the visual application in real time.

Next steps will focus on the relation of maritime emissions, environmental patterns like weather patterns (e.g. wind direction, atmospheric pressure, ...), and air quality on cities, in order to learn which circumstances lead to low quality air scenarios, and how to manage them from a logistic and legal points of view. Further, this technique can also be applied to other scenarios far from maritime modeling, like device networks (known as the Internet of Things) and user modeling for service-providing systems.

[1] http://www.kdd.org/kdd2017

Bibliography

[1] Natalie Mueller, David Rojas-Rueda, Xavier Basagaña, Marta Cirach, Tom Cole-Hunter, Payam Dadvand, David Donaire-Gonzalez, Maria Foraster, Mireia Gascon, David Martinez, et al. Urban and transport planning related exposures and mortality: a health impact assessment for cities. *Environ Health Perspect*, 125:89–96, 2017.

[2] European Community Shipowners Associations (ECSA). The economic value of the eu shipping industry. update. February 2015.

[3] Tristan Smith et al. Third imo greenhouse gas study. 2014.

[4] Francesco Di Natale and Claudia Carotenuto. Particulate matter in marine diesel engines exhausts: Emissions and control strategies. *Transportation Research Part D: Transport and Environment*, 40:166 – 191, 2015.

[5] Mar Viana, Pieter Hammingh, Augustin Colette, Xavier Querol, Bart Degraeuwe, Ina de Vlieger, and John van Aardenne. Impact of maritime transport emissions on coastal air quality in europe. *Atmospheric Environment*, 90:96 – 105, 2014.

[6] A. Soret, M. Guevara, and J.M. Baldasano. The potential impacts of electric vehicles on air quality in the urban areas of barcelona and madrid (spain). *Atmospheric Environment*, 99:51 – 63, 2014.

[7] M. Guevara, F. Martínez, G. Arevalo, S. Gasso, and J. Baldasano. An improved system for modelling spanish emissions: Hermesv2.0. *Atmospheric environment*, 81:209–221, Dec 2013.

[8] N Pérez, J Pey, C Reche, J Cortés, A Alastuey, and X Querol. Impact of harbour emissions on ambient pm10 and pm2.5 in barcelona (spain): Evidences of secondary aerosol formation within the urban area. *The Science of the Total Environment*, 571(5):237–50, 2016-11-15 00:00:00.0.

[9] International Maritime Organization. Ais transponders, February 2017.

[10] J.-P. Jalkanen, a. Brink, J. Kalli, H. Pettersson, J. Kukkonen, and T. Stipa. A modelling system for the exhaust emissions of marine traffic and its application in the Baltic Sea area. *Atmospheric Chemistry and Physics Discussions*, 9(4):15339–15373, 2009.

[11] J. P. Jalkanen, L. Johansson, J. Kukkonen, A. Brink, J. Kalli, and T. Stipa. Extension of an assessment model of ship traffic exhaust emissions for particulate matter and carbon monoxide. *Atmospheric Chemistry and Physics*, 12(5):2641–2659, 2012.

[12] J. P. Jalkanen, L. Johansson, and J. Kukkonen. A comprehensive inventory of ship traffic exhaust emissions in the European sea areas in 2011. *Atmospheric Chemistry and Physics*, 16(1):71–84, 2016.

[13] Miluše Tichavska and Beatriz Tovar. Port-city exhaust emission model: An application to cruise and ferry operations in Las Palmas Port. *Transportation Research Part A: Policy and Practice*, 78(C):347–360, 2015.

[14] Xinjia Gao, Hidenari Makino, and Masao Furusho. Ship behavior analysis for real operating of container ships using ais data. *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation*, 10(2):213–220, 2016.

[15] Ø. Buhaug, J.J. Corbett, Ø. Endresen, V. Eyring, J. Faber, S. Hanayama, D.S. Lee, D. Lee, H. Lindstad, A.Z. Markowska, A. Mjelde, D. Nelissen, J. Nilsen, C. Pålsson, J.J. Winebrake, W.Q. Wu, and K. Yoshida. Second imo ghg study 2009, prevention of air pollution from ships. *International Maritime Organization (IMO)*, 2009.

[16] A. Miola, B. Ciuffo, E. Giovine, and M. Marra. Regulating air emissions from ships. the state of the art on methodologies, technologies and policy options. *Joint Research Centre Reference Report, EUR24602EN, 978-92*, 2010.

[17] Apollonia Miola and Biagio Ciuffo. Estimating air emissions from ships: Meta-analysis of modelling approaches and available data sources. *Atmospheric Environment*, 45(13):2242–2251, 2011.

[18] Enrique Tortosa Solvas and Juan F. Rebollo Lledó. La red AIS portuaria. *Puertos: información mensual de Puertos del Estado*, 158:9–21, 2010.

[19] Albert Bifet and Ricard Gavaldá. Learning from time-changing data with adaptive windowing. In *In SIAM International Conference on Data Mining*, 2007.

[20] Zoubin Ghahramani. Hidden markov models. chapter An Introduction to Hidden Markov Models and Bayesian Networks, pages 9–42. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2002.

[21] Volodymyr Mnih, Hugo Larochelle, and Geoffrey E. Hinton. Conditional restricted boltzmann machines for structured output prediction. In Fábio Gagliardi Cozman and Avi Pfeffer, editors, *UAI*, pages 514–522. AUAI Press, 2011.

[22] Pankaj K. Agarwal and Cecilia M. Procopiuc. Exact and approximation algorithms for clustering. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '98, pages 658–667, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.

[23] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Mining Sequential Patterns by Pattern–Growth:The PrefixSpan Approach. *IEEE Transactions of Knowledge and Data Engineering*, 16(11):1424–1440, 2004.

[24] Christine Parent, Stefano Spaccapietra, Chiara Renso, Gennady Andrienko, Natalia Andrienko, Vania Bogorny, Maria Luisa Damiani, Aris Gkoulalas-Divanis, Jose Macedo, Nikos Pelekis, Yannis Theodoridis, and Zhixian Yan. Semantic trajectories modeling and analysis. *ACM Computing Surveys*, 45(4):42:1–42:32, 2013.

[25] Cepesca Confederación Espa nola de Pesca. El arrastre: Arte de pesca sostenible, February 2017.

[26] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

[27] Graham W. Taylor, Geoffrey E. Hinton, and Sam Roweis. Modeling human motion using binary latent variables. In *Advances in Neural Information Processing Systems*, page 2007. MIT Press, 2006.

[28] E. Forgy. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics*, 21(3):768–769, 1965.

[29] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA*, pages 1027–1035, 2007.

[30] VesselFinder. Free AIS Ship Tracking of Marine Traffic, February 2017.

[31] Achim Zeileis and Gabor Grothendieck. zoo: S3 infrastructure for regular and irregular time series. *Journal of Statistical Software*, 14(1):1–27, 2005.

[32] Jukka-Pekka Jalkanen, Lasse Johansson, and Jaakko Kukkonen. A comprehensive inventory of the ship traffic exhaust emissions in the baltic sea from 2006 to 2009. *AMBIO*, 43(3):311–324, 2014.

[33] W. Qiu and A. Bandara. Gps trace mining for discovering behaviour patterns. In *2015 International Conference on Intelligent Environments*, pages 65–72, July 2015.

[34] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Min. Knowl. Discov.*, 1(3):259–289, January 1997.

[35] Vincent W. Zheng, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative location and activity recommendations with gps history data. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 1029–1038, New York, NY, USA, 2010. ACM.

[36] Zhenhui Li, Jiawei Han, Ming Ji, Lu-An Tang, Yintao Yu, Bolin Ding, Jae-Gil Lee, and Roland Kays. Movemine: Mining moving object data for discovery of animal movement patterns. *ACM Trans. Intell. Syst. Technol.*, 2(4):37:1–37:32, July 2011.

[37] Q. Lin, D. Zhang, X. Huang, H. Ni, and X. Zhou. Detecting wandering behavior based on gps traces for elders with dementia. In *2012 12th International Conference on Control Automation Robotics Vision (ICARCV)*, pages 672–677, Dec 2012.

[38] Asja Fischer and Christian Igel. An introduction to restricted boltzmann machines. In Luis Álvarez, Marta Mejail, Luís Gómez Déniz, and Julio C. Jacobo, editors, *CIARP*, volume 7441 of *Lecture Notes in Computer Science*, pages 14–36. Springer, 2012.

[39] Geoffrey E. Hinton. A practical guide to training restricted boltzmann machines. In Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade (2nd ed.)*, volume 7700 of *Lecture Notes in Computer Science*, pages 599–619. Springer, 2012.

[40] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 791–798, New York, NY, USA, 2007. ACM.

[41] Graham W. Taylor and Geoffrey E. Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1025–1032, New York, NY, USA, 2009. ACM.

[42] Guy Lebanon and S. V. N. Vishwanathan, editors. *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*, volume 38 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2015.

[43] Jaedong Lee, Heera Kim, Noo-ri Kim, and Jee-Hyong Lee. An approach for multi-label classification by directed acyclic graph with label correlation maximization. *Inf. Sci.*, 351(C):101–114, July 2016.