HOBBIT

Collaborative Project

Holistic Benchmarking of Big Linked Data

Project Number: 688227 **Start Date of Project:** 2015/12/01 **Duration:** 36 months

Deliverable 4.2.2
Second Version of the Data Analytics Benchmark

Dissemination Level	Public
Due Date of Deliverable	Month 30, 31/05/2018
Actual Submission Date	Month 30, 31/05/2018
Work Package	WP4 - Benchmarks II: Analysis and processing
Task	T4.2
Type	Other
Approval Status	Final
Version	1.0
Number of Pages	17

Abstract: This deliverable presents the second version of the data analytics benchmark for the HOBBIT Platform.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grand agreement No 688227.

History

Version	Date	Reason	Revised by
0.1	11/05/2018	First draft	Pavel Smirnov
0.2	16/05/2018	Draft revised	Giulio Napolitano
0.3	18/05/2018	Draft revised	Martin Strohbach
1.0	21/05/2018	Final version	Pavel Smirnov

Author List

Organization	Name	Contact Information
AGT International	Pavel Smirnov	psmirnov@agtinternational.com
AGT International	Martin Strohbach	mstrohbach@agtinternational.com
Fraunhofer IAIS	Giulio Napolitano	Giulio.Napolitano@iais.fraunhofer.de

Executive Summary

This document describes the second version of the Data Analytics benchmark, also called Structured Machine Learning (SML) benchmark v2. The benchmark focused on comparative assessment of predictive machine learning algorithms applied to the maritime surveillance domain. The benchmark was used for the ACM DEBS Grand Challenge 2018.

Contents

List of Figures

List of Tables

1 Introduction

Nowadays statistical and machine learning algorithms are highly relevant for remote monitoring and situation-awareness systems [1]. Maritime surveillance is a domain in which analytical algorithms combine varied information originating from heterogeneous data sources and provide a maritime situation picture about remote vessels. Online analysis of maritime situations opens the prospective for solving several industry-relevant tasks, including anomalous behavior detection, insurance risks assessment, supply chain management and logistic planning.

The use case of real time prediction of arrival times and destination ports was chosen for the second version of the Data Analytics benchmark. The goal of the benchmark is to assess the accuracy of various analytical and pre-trained machine learning algorithms, as well as the performance of their implementation using a unified hardware (the HOBBIT platform) and real-time streaming workload. The benchmark itself, as well two sample system implementations (in Java and Python), were used for the ACM DEBS Grand Challenge 2018. The data generator based on the provided (static) dataset was exposed as a set of standalone components, to be used by other benchmarks of the HOBBIT project.

Therefore, the goal of this benchmark is to address the following issues:

- Develop a benchmark based on the task that is representative for analytics software.
- Develop a data generator producing maritime RDF data for other benchmarks of the HOBBIT project.

Table 1 shows our open-source contribution to the project:

Project	URL	Description
Benchmark	https://github.com/hobbit-project/sml-benchmark-v2	Benchmark itself including data generator
Data generator	https://github.com/hobbit-project/sml-v2-datagen	Data generator which provides an RDF stream of marine traffic data (based on the provided static dataset) for other benchmarks
Java System Example	https://github.com/hobbit-project/DEBS-GC-2018	Example of sample system in Java compatible with the benchmark
Python System Example	https://github.com/hobbit-project/python-sample-system	Example of sample system in Python compatible with the benchmark

Table 1: List of open source projects created during the T4.2

The remainder of the document is structured as follows: Section 2 describes the use case, the dataset provided and the required data preprocessing; Section 3 describes the general architecture and some specific details of the benchmark; Section 4 describes the use of the benchmark for the ACM DEBS Grand Challenge 2018.

2 Use case

The section describes the structure of the dataset and use case selected for the benchmark.

2.1 Dataset description

The dataset for the use cases of the benchmark was provided by the MarineTraffic portal [1] and the Big Data Ocean project [2]. The dataset contains anonymized sensor data collected from Automated Identification System (AIS) from a number of ships. AIS is the internationally accepted standard for self-identification of big vessels. AIS transponders periodically send the time-series data points to coastal-, aircraft- or satellite-based AIS receivers. The data points of the dataset provided contain static information (ship identifier, ship type) as well as dynamic data (coordinates, speed, heading, course, draught, departure port name). The dataset is in a comma separated format (csv) and limited by: a) geography - selected ships (more than 500) perform their trips among a predefined number of ports in the Mediterranean Sea; b) timeframe: 3 months. A crossing a port's bounding box is considered as a departure/arrival from/to a port. Bounding boxes or port's radiuses (km) defined manually for each port according to it's distance to other ports in a way to avoid overlapping. The geo coordinates and port's radiuses are provided by a separate CSV file.

Benchmarked analytical algorithms are expected to be trained on 80% and evaluated on 20% of the data respectively.

2.2 Task description

The use case of real time prediction of maritime situations may be splitted into two independent tasks (queries): a) prediction of arrival port names; b) prediction of arrival times.

In order to make the use case scenario more realistic, the benchmark implements the sequential processing logic over each of the several data streams. All ships from a dataset can be processed in parallel but data points of each particular ships should be processed sequentially: each new data point should be sent to the benchmarked algorithm/system only after the prediction from the previously sent data points has returned back. This logic reproduces real monitoring systems, which receive raw data in real time manner and should compute analytics immediately.

Evaluated algorithms (benchmarked systems) receive data points (data tuples) with the following format:

SHIP_ID - the anonymized id of the ship

SHIPTYPE - defined according to the reference [3]

SPEED - measured in knots (value divided by 10)

LON - the longitude of the current ship position

LAT - the latitude of the current ship position

[1] http://marinetraffic.com
[2] http://www.bigdataocean.eu
[3] http://bit.ly/2IIsSHS

COURSE - the direction in which the ship moves [4]

HEADING - defined according to [4]

TIMESTAMP - time at which the message was sent (UTC)

DEPARTURE_PORT_NAME - name of the last port visited by the vessel

REPORTED_DRAUGHT - vertical distance between the waterline and the bottom of the hull (keel) [5]

2.3 Evaluation

Depending on the task specified in configuration parameters, the benchmark should expect different results from the benchmarked system and perform the calculation of different KPIs.

The first task (Query 1) considers an arrival port name as the prediction result. For the Query 1 the KPI is an average earliness rate of given predictions. The earliness rate is calculated per trip and refers to the proportion between lengths of (1) last correct predictions sequence and (2) sequence of all tuples of the trip (see Fig. 1). A group of consecutive data points between two ports (one of them is stated as DEPARTURE_PORT_NAME) with the same SHIP_ID is considered as trip.

Finally, the average earliness rate for all trips is returned as a KPI (see Fig. 2)):

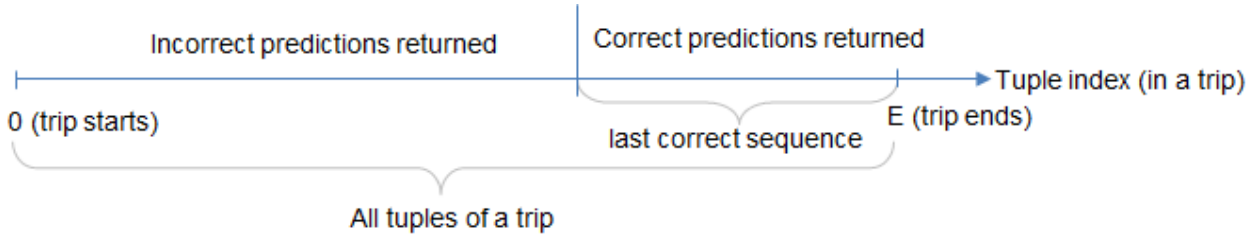

Figure 1: Schematic representation of evaluated entities for the Query 2.

$$A = \frac{\sum_{k=0}^{Ntrips} \frac{len(last\ correct\ sequence)}{len(all\ tuples\ of\ a\ trip)}}{Ntrips}$$

Figure 2: Formal definition of the earliness rate KPI (Query 1).

The second task (Query 2) considers the arrival timestamp as the prediction result. For the Query 2 the mean absolute error (minutes) is considered as KPI. It is calculated as a sum of absolute deviations between predicted and actual arrival timestamps for all tuples sent to system (see Fig. 3, 4).

[4] https://en.wikipedia.org/wiki/Course_(navigation)
[5] https://en.wikipedia.org/wiki/Draft_(hull)

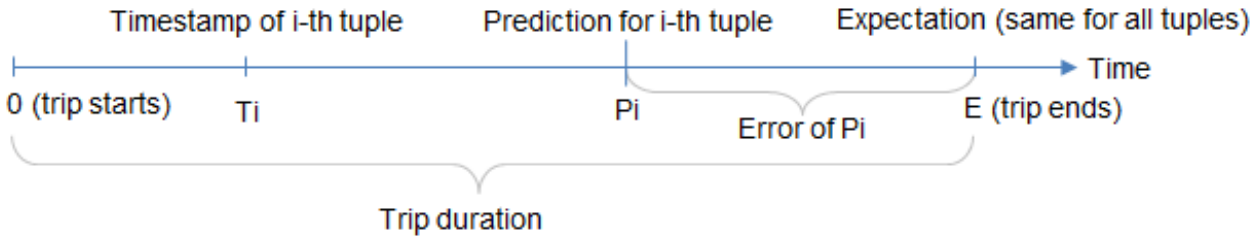

Figure 3: Schematic representation of evaluated entities for the Query 2.

$$A = \frac{\sum_{i=0}^{Ntuples} abs(error\ per\ tuple)}{Ntuples}$$

Figure 4: Formal definition of the mean absolute error KPI (Query 2).

2.4 Ground truth calculation

In order to have a valid ground truth dataset for benchmarked system evaluation, prior data processing is required. The preprocessing has two goals:

1. Grouping of all data points into individual ship trips.
2. Labelling of obtained trips.

While the provided dataset already contains the required anchor values (departure port name, arrival port name, arrival timestamp) to enable grouping and labelling, manual inspection of the processed data highlighted a number of problems:

1. Actual geo coordinates of the ships not always demonstrate correlation between stated departure and arrival ports. Some ships perform several back and forth trips between several ports, but all data points of the trip have the same anchor values so they look like a single trip with long duration.
2. Ships may arrive to some third-party port not listed in the ports list. While the dataset was requested to be limited to only 25 ports, visual inspection of the coordinates of long parking times outside of radiuses of the ports listed demonstrated the fact that many ships visited third party ports without making any trip-related (departure port name, departure time) changes in AIS.
3. Ships may be parked for some weeks or months and continue to send data points with anchor values. This gives incorrect arrival timestamps, so parking should be excluded since we have no information about reasons of the parking episodes to predict them.
4. AIS data may be missing for several hours or even days during a trip. This might be considered as a technical or a human factor problem and highly impact on a accuracy of the assessment.

The outlined problems demonstrated that the anchor values provided are not reliable for the correct evaluation. Thus a set of algorithms to deal with these data issues was developed. The algorithms perform the following transformations over the provided data:

1. Find parking points. The sliding window algorithm was applied to detect continuous sequences of tuples, where window-aggregated speed is below 1 knots. Parking groups allowed to split trips by long (>5% of whole duration) parking times and not consider them to be labelled at least (if none of the next transformations occur).
2. Find the ports not listed - by a visual inspection (see Fig.5).
3. Recalculate the real arrival and departure ports using geo coordinates of ships and ports.

The described measures allowed to perform labeling for more than 80% of the is original data as well as to sort visualizations of labelled/not labelled trips for visual inspection of correctness.

Figure 5: Example of trips splitted by parking times: two trips have been splitted to 4 trips, parking times in ports have been excluded from the assessment.

3 Benchmark

The benchmark is performed in a streaming fashion. That is, the benchmark is able to generate data, evaluate benchmarked system's output and compute assessment metrics at same time. This section describes the general architecture as well as some implementation details of the benchmark.

3.1 Architecture

The architecture of the benchmark is shown in Fig.6. The benchmark is built using the standard HOBBIT's benchmarks architecture [6] with some optimizations. Since the benchmark is oriented to work in streaming manner, some optimizations have been done in order to achieve the best performance. First, since the benchmark does not require to preload any data into a benchmarked system, a standard Data Generator component was removed from the architecture and data generation was implemented in the Task Generator component. Second, since the benchmark requires a custom in-memory Evaluation Storage with an acknowledgement mechanism (described above), it provided an opportunity to integrate an Evaluation Module directly into the Evaluation Storage and to remove another redundant stream. The role of the components as well as the implemented optimizations are described below.

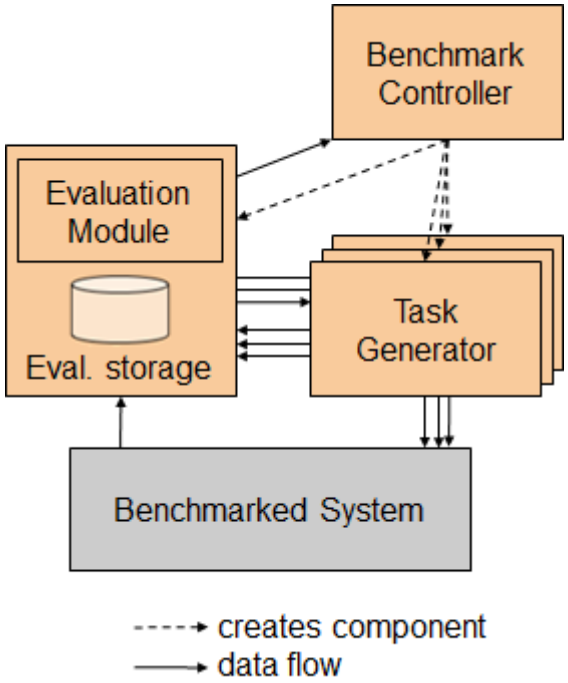

Figure 6: Example of trips splitted by parking times: two trips have been splitted to 4 trips, parking times in ports have been excluded from the assessment.

Benchmark Controller. The Benchmark Controller is the main component, which initiates the deployment of other components and controls the benchmarking process. The Benchmark Controller uses platform-compatible protocols and communicates with the Command Queue of the platform. The benchmark controller is able to initiate multiple instances of the Task Generator component in order to increase the rate of input stream for a benchmarked system.

Task Generator. The Task Generator is a component intended to emit data points (tuples) into an exchange queue, from which data tuples will be consumed by a benchmarked system. The task generator reads the provided dataset file, extracts the corresponding part of data (in case of a benchmark run with multiple Task Generator instances) and generates a data tuple with a unique identifier (task), which will be sent to the benchmarked system. Data tuples for each ship are send sequentially, i.e. each new tuple for a ship will be send only after the evaluation receives

[6] https://project-hobbit.eu/wp-content/uploads/2018/03/D2.2.2_Second_Version_of_the_HOBBIT_Platform.pdf

the result of the previous tuple processing and sends a notification to a data generator. After the benchmarked system has finished processing all the tuples, the Task Generator component starts to send an expected processing results (ground truth) for each data point to Evaluation Storage. Such a delayed sending guarantees that the ground truth data will not be captured by a benchmarked system and used for more precise results calculation.

Evaluation storage. The Evaluation Storage inherits the standard in-memory evaluation storage implementation and extends it with an acknowledgement mechanism. From a benchmarked system it receives a response (processing results) associated with particular task identifier and sends an acknowledgments with encrypted task identifiers to the Task Generator component. Acknowledgments with encrypted task identifiers guarantee that Task Generator is secured from fake produced notifications (e.g. produced by the benchmarked system).

Evaluation Module. The Evaluation Module component implements the default HOBBIT's implementation, but its functionality is called from the Evaluation Storage component. The Evaluation Module compares the "expected-actual" pairs for each task and calculates the benchmark KPIs according to the requirements of the configuration.

The interaction between the components is represented by the diagram in Fig 7.

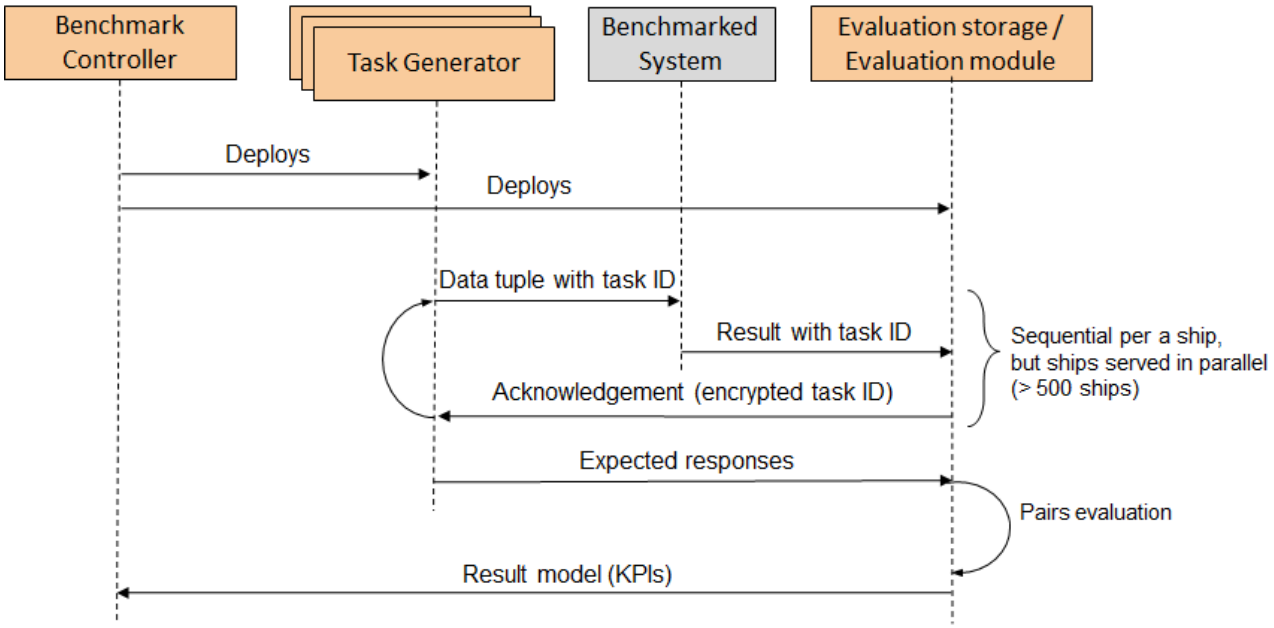

Figure 7: Interaction between benchmark components.

3.2 Input parameters

The benchmark can handle the following configuration parameters:

Tuples limit - service parameter (relevant for debugging), allows to limit the maximum number of data points to be sent to the benchmarked system. Unlimited by default.

Benchmark timeout - service parameter, allows to limit the maximum execution time. Unlimited by default.

Query type - required parameter, defines the benchmark task and relevant KPIs calculation.

Data format - optional, allows to select between CSV and RDF format (see below).

3.3 Data formats

In order to provide data in RDF format the datAcron ontology [7] was selected. The ontology contains classes and properties for describing trajectories and events of moving objects in the maritime and air traffic domain. Fig. 8 depicts corresponding classes of the Dat Acron ontology, which have been selected to represent marineTraffic data.

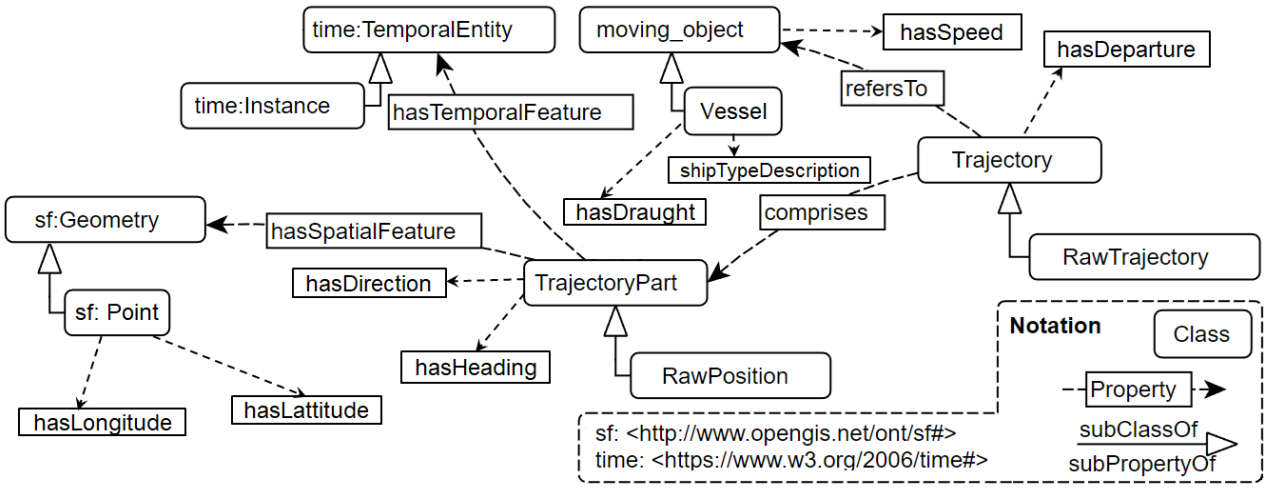

Figure 8: Classes of the Dat Acron ontology used for the RDF representation.

The equivalence of data representation in two formats (CSV and a RDF) is shown in Fig. 9.

3.4 Measured KPIs

The benchmark measures the following performance indicators (KPIs) shown in Fig.10:

Average earliness rate (%) - how early from the end system/algorithm started to provide correct port name predictions until the end of trip (calculated for the Query 1 only). The formal definition of the KPI is in Section 2.

Mean absolute error (min) - total absolute deviation of predicted arrival times from the actual arrival times (calculated for the Query 2 only). The formal definition of the KPI is in Fig. 2 and Fig. 4.

Average processing latency (ms) - the time (ms) the system spends for a single prediction.

System working time (seconds) - the absolute difference between timestamps of first task sent to system and a timestamp of the last received response from system.

Evaluated pairs count - amount of correctly formatted responses returned from system, but limited with a number of labeled data points.

[7] http://ai-group.ds.unipi.gr/datacron_ontology

SHIP_ID	SHIPTYPE	SPEED	LON	LAT
0xc35c9ebbf48cbb5857a868ce441824d0b2ff783a	99	8.2	14.56	35.81

COURSE	HEADING	TIMESTAMP	DEPARTURE_PORT_NAME
109	511	10-03-15 12:15	MARSAXLOKK

REPORTED_DRAUGHT	TRIP_ID
	0xc35c9_10-03-15 12:xx - 10-03-15 13:26

RDF

```
<datAcron:Trajectory rdf:about="http://project-hobbit.eu/sml-benchmark-v2/trajectory#0xc35c9_10-03-15  12:xx - 10-03-15 13:26">
    <datAcron:refersTo>
      <datAcron:Vessel rdf:about="http://project-hobbit.eu/sml-benchmark-v2/ship#0xc35c9ebbf48cbb5857a868ce441824d0b2ff783a">
        <datAcron:hasSpeed>8.2</datAcron:hasSpeed>
        <datAcron:has_draught rdf:datatype="http://www.w3.org/2001/XMLSchema#int">none</datAcron:has_draught>
        <datAcron:shipTypeDescription>99</datAcron:shipTypeDescription>
      </datAcron:Vessel>
    </datAcron:refersTo>
    <datAcron:hasDeparture>MARSAXLOKK</datAcron:hasDeparture>
</datAcron:Trajectory>
```

Figure 9: Relation between CSV and RDF representation (RDF triples are presented partly).

4 Using the Benchmark in ACM DEBS Grand Challenge 2018

The developed benchmark was used as the basis for the DEBS Grand Challenge 2018 [8]. The DEBS Grand Challenge is an annual event processing challenge series co-organized with the ACM Distributed Event-Based Systems conference (DEBS). The scope of the challenge series is to evaluate a variety of analytical solutions focused on the same problem from a particular event/stream processing domain. Last year challenge were focused on Industry 4.0 and was organized on the HOBBIT platform with first version of Data Analytics Benchmark [9]. Since the DEBS GC 2018 is more oriented towards the machine learning community, as opposed to the RDF stream processing one, it was decided to use the csv format for the Challenge.

4.1 Leaderboards

In the online platform the DEBS GC 2018 was organized as two consecutive challenges phases: the training and the final one. The training phase challenge was intended to demonstrate and ranking of participating systems using a small part of the test set. The training challenge was specified to be executed periodically and results have been updated on the publicly available challenge leaderboards (see Fig.11, names of participated systems are hidden, because the training phase is not yet finished). The final phase challenge is intended use all the test dataset and to be executed after the training phase is over. The final results (leaderboard) will be not publicly available until the awarding procedure at the DEBS 2018 conference (25-29 June 2018).

[8] http://www.cs.otago.ac.nz/debs2018/calls/gc.html

[9] https://project-hobbit.eu/wp-content/uploads/2017/06/D4.2.1_First_version_of_the_data_analytics_benchmark.pdf

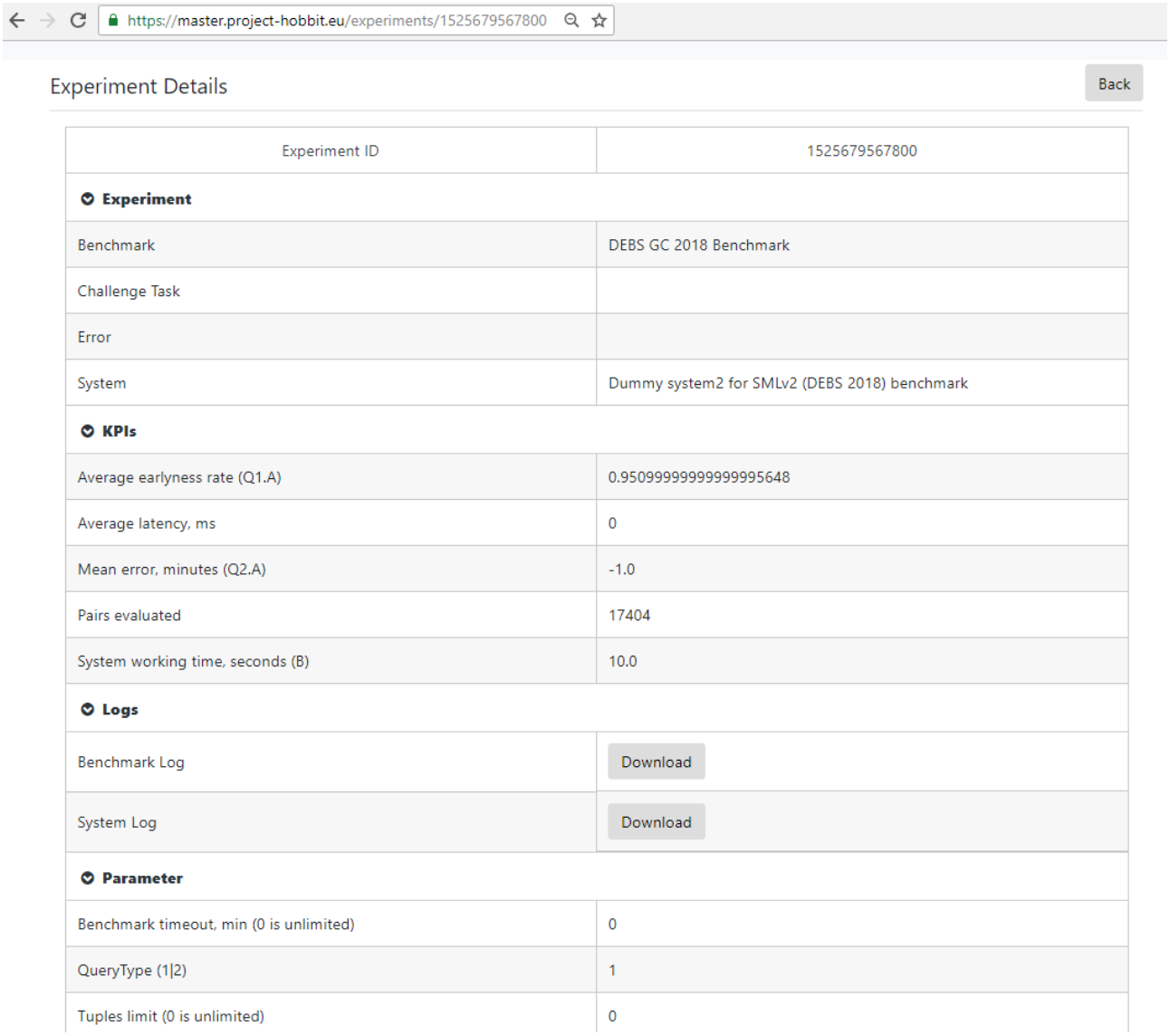

Figure 10: Screenshot of the benchmark execution results.

4.2 Sample systems

Two sample systems on Java and Python were developed and published in order to ease the integration with the HOBBIT platform for challenge participants. The sample systems are using the proposed JAVA SDK [10] and use the publicly available benchmarks' docker images from the online platform, which allowed participants to debug their systems locally fully reproducing the execution process of the online platform. More details about the sample systems are described in the corresponding repositories [11][12].

[10] https://project-hobbit.eu/wp-content/uploads/2018/03/D2.2.2_Second_Version_of_the_HOBBIT_Platform.pdf

[11] https://github.com/hobbit-project/DEBS-GC-2018

[12] https://github.com/hobbit-project/python-sample-system

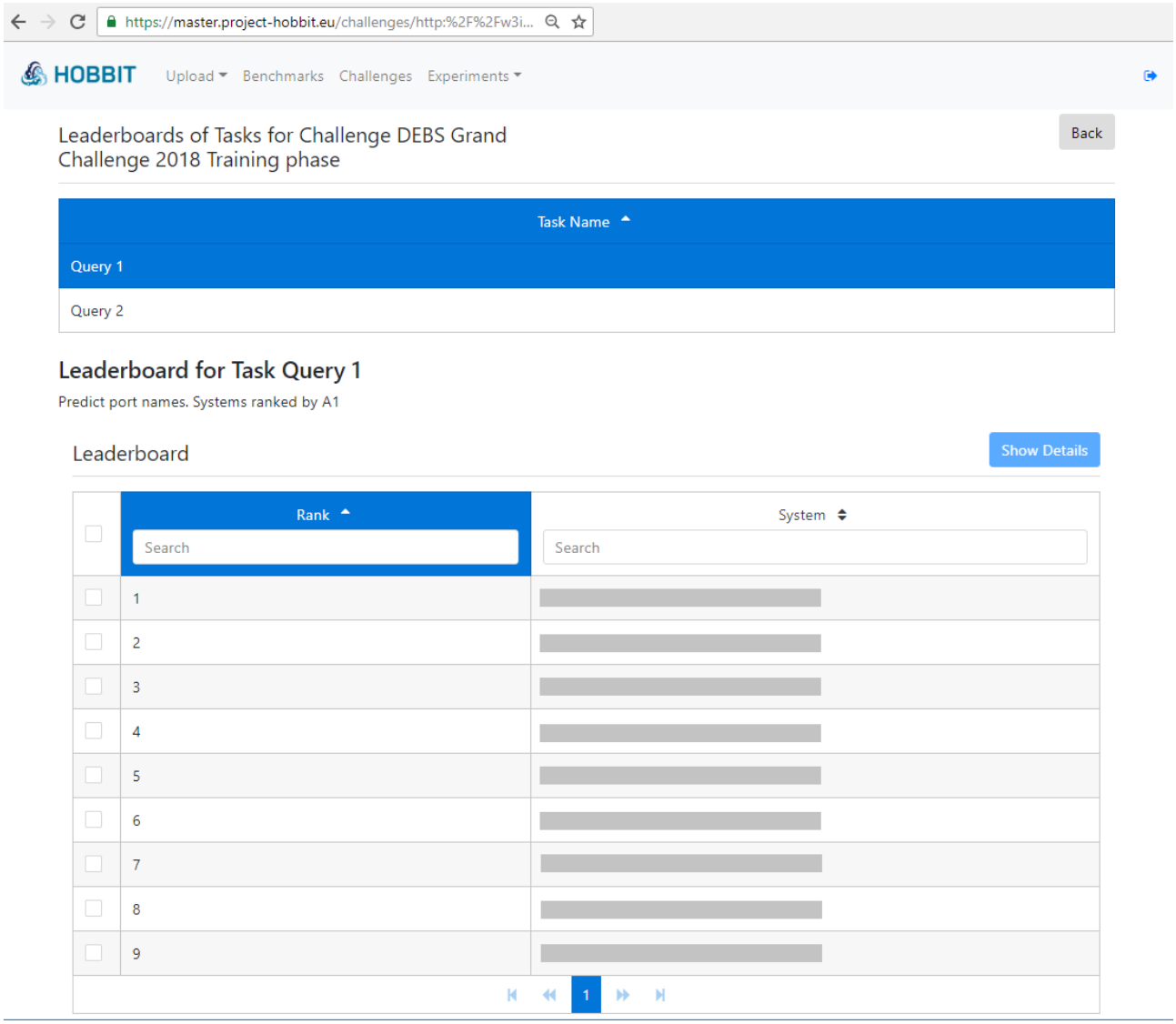

Figure 11: Screenshot of the the training phase leaderboard (names of participating systems are hidden).

4.3 Issues tracking

In order to help the participants to solve integrational problems, an online issue tracker at the project Github [13] was announced. The tracker made possible to answer questions about the platform, benchmark and challenge in a shorter time. At the moment the tracker is filled with more than 30 questions and discussions.

5 Summary

The second version of the data analytics benchmark was described in the document. The benchmark allows to assess and compare predictive capabilities of machine learning algorithms in the maritime

[13] https://github.com/hobbit-project/DEBS-GC-2018/issues

situation awareness domain. The benchmark and the HOBBIT platform was chosen by the ACM Grand Challenge organizers as the technical solution for performing the challenge. This fact demonstrates that the developed benchmark, as well as the HOBBIT platform (for the second time), adequately addresses the benchmarking task in analytics domains.

References

[1] Mica R Endsley and Daniel J Garland. *Situation awareness analysis and measurement*. CRC Press, 2000.