Gradual Pattern Mining Tool on Cloud

Dickson Owuor*,**, Anne Laurent*
Joseph Orero**, Olivier Lobry***

*LIRMM Univ Montpellier, CNRS, Montpellier, France
doowuor@lirmm.fr
anne.laurent@umontpellier.fr
**FIT Strathmore University, Nairobi, Kenya
{dowuor, jorero}@strathmore.edu
***OSU OREME Univ Montpellier, Montpellier, France
olivier.lobry@umontpellier.fr

Résumé. This paper describes an approach that illustrates how gradual pattern mining algorithms are integrated into a containerized *Docker* Cloud platform that implements *OGC* (Open Geospatial Consortium) SensorThings API (application interface). We present a practical application of the SensorThings API as a source of real-time data streams and propose an architecture that allows for extraction of gradual patterns among these data streams.

1 Introduction

Scientific researchers are constantly collecting, crossing and analyzing data in order to help them understand various phenomena. For example environmental data is important for helping to understand phenomena like global warming, rainfall patterns etc. Most of such data is collected using sensors that are enabled to upload data in Cloud frameworks (Liang et al., 2016; Hajj-Hassan et al., 2018; Joshi et Simon, 2018). These researchers obviously wish to spend more time studying phenomena than configuring research tools. Fortunately, integrating data analysis tools into Cloud frameworks alleviates such configuration hustles.

It is important to highlight that understanding certain phenomena may require knowledge to be extracted from numerous unrelated data sets. The technique of combining unrelated data sets in order to identify interesting correlations, also known as *data crossing*, is increasingly gaining more interest in numerous research domains (Hajj-Hassan et al., 2018).

Temporal gradual pattern (GP) mining is an instance of a data analysis technique that allows for extraction of gradual correlations among attributes of a data set. For instance given a set of an ordered time-series data set with attributes {Temperature, Mosquitoes}, a temporal GP may take the form : *"the higher the temperature, the more mosquitoes almost 2 hours later"* (Owuor et al., 2019). Time-series data sets that are collected in real-time are also known as *data streams*. The OGC SensorThings API is an example of a framework that generates and stores data streams from numerous environmental sensors.

The idea of a software implementation that enables the access of GP mining algorithms through a SaaS (Software-as-a-service) model through a Web interface is interesting. In a SaaS

distribution model, software is centrally hosted and licensed on a subscription basis. This introduces a flexibility that spares users the agony of spending hours trying to install analysis software (Joshi et Simon, 2018).

It is an added advantage if the software implementation allows users to select and cross numerous unrelated data streams into a single data set from which temporal GPs can be mined. For instance a user can cross a *"temperature"* data stream with a *"no. of bees"* data stream to test for a pattern like : *"the higher the temperature, the higher the population of bees almost 3 days later"*.

According to (Owuor et al., 2019) the computational complexity of mining GPs increases with the size of the data set. For this reason, the algorithms are implemented in Python language in order to harness Python's ability to improve efficiency through multi-core parallel processing. The proposed SaaS allows implementations of these algorithms on a powerful supercomputer and provides a user-friendly client interface for the users.

In this paper, we propose a software architecture model (on a `Docker` Cloud platform) that implements a crossing model for time-series data proposed by (Owuor et al., 2020) and integrates *T-GRAANK* algorithm proposed by (Owuor et al., 2019) in order to allow for extraction of GPs from crossed data streams. The remainder of this paper is organized as follows : we describe the OGC SensorThings API in Section 2 ; we describe the proposed model in Section 3 ; finally, we conclude and give future directions regarding this work in Section 4.

2 OGC SensorThings framework

(Open Geospatial Consortium) OGC [1] is an international consortium consisting of over 530 companies, government agencies, research organizations and universities driven to make geospatial data and services *FAIR* (Findable, Accessible, Interoperable and Reusable). The OGC SensorThings API is an open standard that is built on top of the OGC Sensor Web Enablement (SWE) and (International Organization for Standardization) ISO/OGC Observation and Measurement (O&M) data model (Van de Crommert et al., 2004).

The SensorThings API is specifically designed for constrained *IoT* (Internet of Things) devices ; therefore, it employs the use of efficient technologies such as RESTful (Representational State Transfer) API, JSON (JavaScript Object Notation) encoding, MQTT protocol, OASIS OData (Open Data protocol) and flexible URL conventions (Liang et al., 2016). The OGC SensorThings API is composed of 2 parts : (1) the *Sensing* part and (2) the *Tasking* part. In this paper, we implement the *Sensing* part which allows IoT devices and applications to perform CRUD operations through HTTP requests (i.e. `POST, GET, PATCH, DELETE`).

The *Sensing* part is designed based on the ISO/OGC O&M data model and it defines 8 entities for IoT sensing applications. Figure 1 depicts the 8 entities together with their properties and relationships (Hajj-Hassan et al., 2015; Liang et al., 2016). When implemented, the OGC SensorThings exposes a service document resources, that lists the 8 entity sets, that allows its clients to navigate the entities in a hyper-media driven manner.

1. `https://www.opengeospatial.org/`

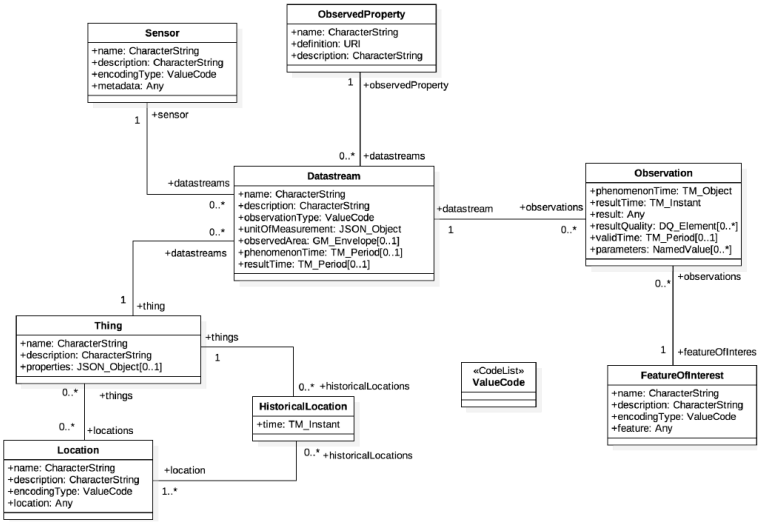

FIG. 1 – *The sensing entities that make the OGC SensorThings API*

3 Proposed model

3.1 System architecture

We propose to extend GOST [2] (golang SensorThings), a certified software implementation of the OGC SensorThings API, to allow for extraction of gradual patterns from unrelated data streams as illustrated in Figure 2.

We implement our model on top of the OGC SensorThings API because, to the best of our knowledge, it stands out as the best API to interconnect IoT devices, sensor data and applications over the Cloud. As can be deduced in Figure 2, the OGC Sensor Things API implementation provides : (1) OGC/SensorThings service, (2) MQTT component for sensor data collection and, (3) data stream storage component. We extend the API and include :

1. data crossing software component - which crosses numerous data streams into one data set and,
2. gradual pattern mining software component - which extracts GPs from the crossed data set.

2. `https://www.gostserver.xyz`

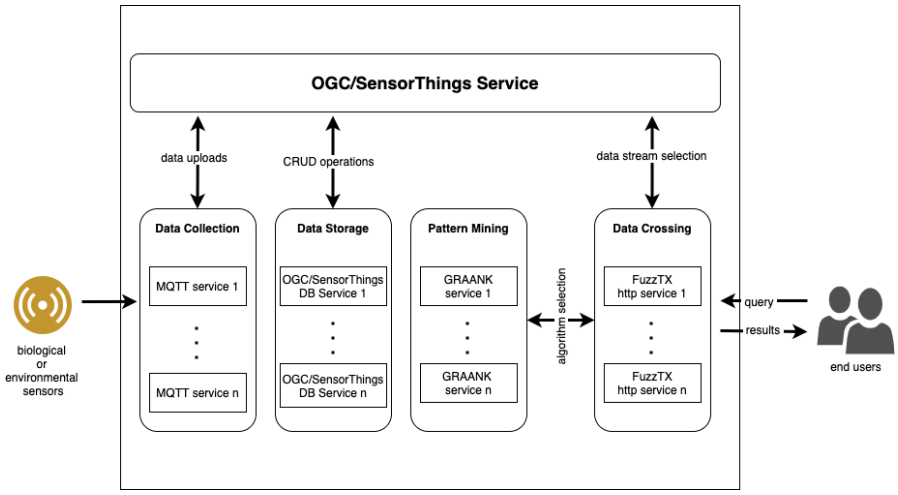

FIG. 2 – *Proposed system architecture*

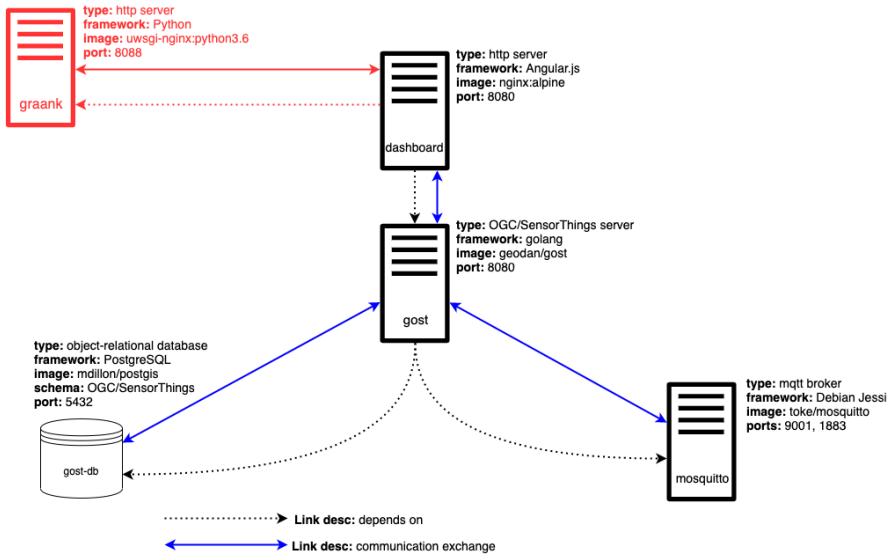

FIG. 3 – *Docker architecture of a Cloud API for integrating GP mining algorithms into a Docker implementation of the OGC SensorThings API*

As shown in Figure 2 and Figure 3, the proposed system is composed of 5 items which we described below.

(1) OGC SensorThings Service (implemented by **GOST** server) exposes the document resources of the 8 entity sets (listed in Section 2) for the SensorThings clients. This is the core

task of the SensorThings API since every CRUD action go through it.

(2) Data Collection tool implements MQTT protocol which is a lightweight publish-subscribe protocol designed for constrained devices. This tool (implemented by **Mosquitto** server) is used to connect to sensors in order to collect data from the environment and send them to *OGC SensorThings Service*.

(3) Data Storage tool (implemented by **GOST database**) serves an object-relational database which implements the schema of the 8 OGC SensorThings sensing entities.

(4) Data Crossing tool serves a HTTP **Dashboard** which enables users to interact with our system. Through this tool users are able to send HTTP requests to the *OGC SensorThings Service* and the *Pattern Mining tool*. Specifically, this tool implements `FuzzTX` algorithm proposed by (Owuor et al., 2020) which allows for selection and crossing of numerous data streams into one data set. It sends the crossed data set to the `Pattern Mining tool` for extraction of gradual patterns and presents the extracted patterns to the user.

(5) Pattern Mining tool (implemented by **GRAANK** server) serves the GP mining algorithms which are variants of the `T-GRAANK` (Temporal GRAdual rANKing) technique proposed by (Owuor et al., 2019) and are implemented in Python. This tool runs services that receive HTTP $POST$ requests from the *Data Crossing tool* and responds with the extracted patterns.

3.2 Demonstration

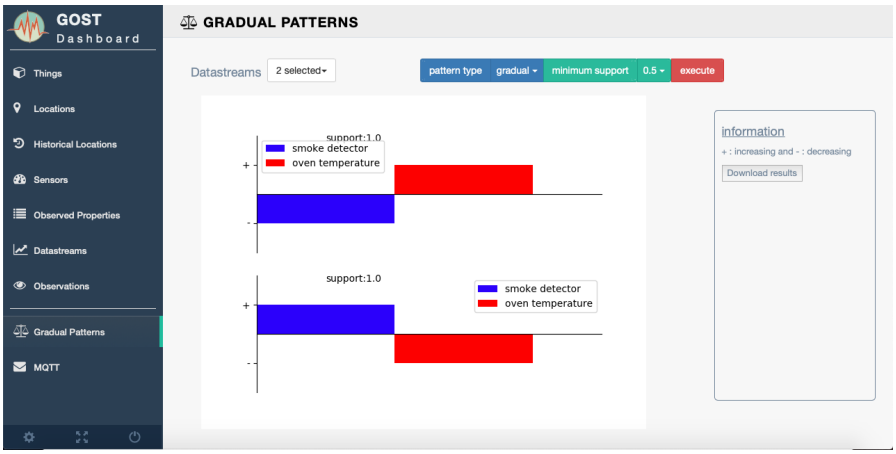

FIG. 4 – *Screenshot of implemented Cloud application for demonstration*

Our proposed Cloud application mostly targets users that require to cross unrelated time-series data collected by sensors and mine them for patterns or knowledge that is not obvious. For demonstration purposes, we allow the application to mine for gradual patterns from crossed sensor data. We proceed to describe how users may interact with the system which is designed to run on a Web interface which is shown in Figure 4.

The left panel of the Web application provides navigation items for accessing : “Things”, “Locations”, “Historical Locations”, “Sensors”, “Observed Properties”, “Datastreams”, “Ob-

servations", "Gradual Patterns" and "MQTT". The right panel provides an interface for any navigation item selected by the user. The user clicks on the "Things" navigation item to add a phenomena that he/she is interested in observing. The phenomena (or thing) allows the user to add define data streams, locations, and sensors that will collect data. The user can view these properties by clicking on "Datastreams", "Locations" and, "Sensors".

In order to cross different data streams and mine them for gradual patterns, the user clicks on the "Gradual Pattern" navigation item and the interface on the right panel will appear as shown in Figure 4. At the top of this panel, there are two configuration settings :

— first one allows the user to choose which data streams to cross and,
— second one allows the user to specify the type and quality of the gradual patterns to be extracted.

In this demonstration, the user selects "smoke detector" data stream and "oven temperature" data stream and specifies extraction of gradual patterns whose support is higher than 0.5. Below the configuration settings, is the results interface and it displays two complementary patterns that give the same knowledge. The gradual pattern can be interpreted as *"the higher the oven temperature, the less the smoke" or vice-versa*. We point out that the data streams used to mine this pattern are synthetic for the purposes of demonstration and they do not reflect any real scenario.

3.3 Possible application area

OREME (Observatory of Mediterranean Research of the Environment) is a scientific observatory that complies with INSPIRE [3] *Implementing Rules* and supports research activities that involve continuous observation of nature over long periods in the Mediterranean region.

Currently, OREME provides OGC geospatial standards like WFS (Web Feature Service) and WMS (Web Map Service) through its data portal [4] ; however, it has not yet implemented data stream standards like the OGC SensorThings due to data entity incompatibility. One solution may involve installing IoT sensors at the data collection stages, which integrate into an MQTT instance to generate compatible data streams (Grellet et al., 2017; Kotsev et al., 2018).

4 Conclusions and future works

In this paper, we propose and develop a software architecture model that allows gradual pattern mining algorithms to be accessed over a Cloud platform. This allows for these algorithms to be applied on real-time sensor data that is collected over the Cloud through frameworks such as the OGC SensorThings. In future, we intend to deploy this system on top of active research observatories such as the OREME's data portal.

Availability of materials. The entire source code for this work is available at our *GitHub repository* : `https://github.com/DAJOOL2020/cloud-api.git`.

3. `https://inspire.ec.europa.eu`
4. `https://data.oreme.org`

Références

Grellet, S., M. Beaufils, K. Schleidt, A. Sarretta, P. Tagliolato, S. Jirka, O. Alessandro, J. M. Rubio Iglesias, et A. Kotsev (2017). Workshop : Integration of O&M data in the INSPIRE SDI - Benefits, challenges and prospects. In *INSPIRE Conference 2017*, Kehl, Germany.

Hajj-Hassan, H., N. Arnaud, L. Drapeau, A. Laurent, O. Lobry, et C. Khater (2015). Integrating sensor data using sensor observation service : Towards a methodology for the o-life observatory. *Sensors & Transducers 194*(11), 99.

Hajj-Hassan, H., A. Laurent, et A. Martin (2018). Exploiting inter- and intra-base crossing with multi-mappings : Application to environmental data. *Big Data and Cognitive Computing 2*(3).

Joshi, O. S. et G. Simon (2018). Sentiment analysis tool on cloud : Software as a service model. In *2018 International Conference On Advances in Communication and Computing Technology (ICACCT)*.

Kotsev, A., K. Schleidt, S. Liang, H. Van der Schaaf, T. Khalafbeigi, S. Grellet, M. Lutz, S. Jirka, et M. Beaufils (2018). Extending INSPIRE to the Internet of Things through SensorThings api. *Geosciences 8*(6).

Liang, S., C.-Y. Huang, et T. Khalafbeigi (2016). OGC SensorThings API part 1 : Sensing, version 1.0.

Owuor, D., A. Laurent, et J. Orero (2019). Mining fuzzy-temporal gradual patterns. In *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, New York, NY, USA, pp. 1–6. IEEE.

Owuor, D. O., A. Laurent, et J. O. Orero (2020). Exploiting IoT data crossings for gradual pattern mining through parallel processing. In *ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium*, Cham, pp. 110–121. Springer International Publishing.

Van de Crommert, P., F. Langelaan, et J. van Winden (2004). OGC web services in action. *Geo-information Standards in Action*, 67–72.

Summary

This paper describes a software architecture model that integrates temporal gradual pattern (GP) mining algorithms into a Cloud platform which implements *OGC* (Open Geospatial Consortium) SensorThings API (application interface). We build the model on top of the SensorThings API in order to exploit the data streams it generates for extraction of temporal GPs.