Environmental Modelling & Software 58 (2014) 27–47

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Representing situational knowledge acquired from sensor data for atmospheric phenomena

Markus Stocker [a,*], Elham Baranizadeh [b], Harri Portin [b,d], Mika Komppula [d], Mauno Rönkkö [a], Amar Hamed [b], Annele Virtanen [b], Kari Lehtinen [b,d], Ari Laaksonen [b,c], Mikko Kolehmainen [a]

[a] Environmental Informatics Group, Department of Environmental Science, University of Eastern Finland, P.O. Box 1627, 70211 Kuopio, Finland
[b] Aerosol Physics Group, Department of Applied Physics, University of Eastern Finland, P.O. Box 1627, 70211 Kuopio, Finland
[c] Finnish Meteorological Institute, P.O. Box 503, 00101 Helsinki, Finland
[d] Finnish Meteorological Institute, P.O. Box 1627, 70211 Kuopio, Finland

ARTICLE INFO

Article history:
Received 17 October 2013
Received in revised form
10 February 2014
Accepted 3 April 2014
Available online 8 May 2014

Keywords:
Environmental sensor networks
Sensor data
Knowledge acquisition
Knowledge representation
Atmospheric phenomena

ABSTRACT

A recurrent problem in applications that build on environmental sensor networks is that of sensor data organization and interpretation. Organization focuses on, for instance, resolving the syntactic and semantic heterogeneity of sensor data. The distinguishing factor between organization and interpretation is the abstraction from sensor data with information acquired from sensor data. Such information may be situational knowledge for environmental phenomena. We discuss a generic software framework for the organization and interpretation of sensor data and demonstrate its application to data of a large scale sensor network for the monitoring of atmospheric phenomena. The results show that software support for the organization and interpretation of sensor data is valuable to scientists in scientific computing workflows. Explicitly represented situational knowledge is also useful to client software systems as it can be queried, integrated, reasoned, visualized, or annotated.

Software availability

Wavellite is open source, is written in Java, and was released in 2013 under the Eclipse Public License (EPL 1.0). Wavellite was developed and is maintained by Markus Stocker. The Wavellite project page is at http://www.uef.fi/en/envi/projects/wavellite.

1. Introduction

Environmental sensor networks are an important research tool for earth and environmental science (Hart and Martinez, 2006). They play a key role in the monitoring of the natural environment and allow for unprecedented study of the dynamics of environmental systems and processes (Hill et al., 2011).

Over the past decades, many small and large scale environmental sensor networks have been deployed. The Finnish Station for Measuring Ecosystem-Atmosphere Relations (SMEAR) is an example for a large scale sensor network and Hart and Martinez (2006) present several others. SMEAR started its operations in 1991 with measurements for SO_2 in Eastern Lapland. It quickly grew to include several other locations and properties of environmental phenomena, including for weather, such as temperature, humidity, or wind speed; for atmospheric gases, such as the concentration of carbon dioxide or ozone; for aerosols, such as particle number concentration.

Today, SMEAR consists of four main stations: SMEAR I in Eastern Lapland, SMEAR II in Hyytiälä, SMEAR III in Helsinki, and SMEAR IV in Kuopio. The main stations consist of one or more substations. Substations consist of a set of sensing devices. For instance, SMEAR IV consists of two substations, at Puijo and at Savilahti. The substation SMEAR IV-Puijo resides on top of the Puijo observation tower (62°54′32″ N, 27°39′31″ E), 306 m above sea level and 224 m above the surrounding lake level. The Puijo observation tower is located in the city of Kuopio, in a semi-urban environment. Kuopio is situated in Eastern Finland, about 330 km to the northeast from Helsinki. SMEAR IV-Puijo consists of sensing devices for the monitoring of aerosols, weather, and atmospheric gases. Sensing devices are manufactured by various vendors, including Thermo Fisher Scientific Inc., TSI Inc., and Vaisala (Leskinen et al., 2009). In this study, we used data by sensors of SMEAR IV-Puijo.

* Corresponding author. Tel.: +358 44 500 3908; fax: +358 17 163 191.
E-mail addresses: markus.stocker@uef.fi (M. Stocker), elham.baranizadeh@uef.fi (E. Baranizadeh), harri.portin@fmi.fi (H. Portin), mika.komppula@fmi.fi (M. Komppula), mauno.ronkko@uef.fi (M. Rönkkö), amar.hamed@uef.fi (A. Hamed), annele.virtanen@uef.fi (A. Virtanen), kari.lehtinen@fmi.fi (K. Lehtinen), ari.laaksonen@fmi.fi (A. Laaksonen), mikko.kolehmainen@uef.fi (M. Kolehmainen).

http://dx.doi.org/10.1016/j.envsoft.2014.04.006
1364-8152/

Environmental sensor networks can produce large amounts of syntactically and semantically heterogeneous data. SMEAR IV-Puijo alone generates approximately 2.5 million data points every day, of which 1 million are by a single sensor (namely the optical cloud droplet spectrometer). To ensure its utility, such data must be managed with appropriate hardware and software systems (Hart and Martinez, 2006; Horsburgh et al., 2009). We distinguish the class of software systems that organize sensor data and the class of software systems that in addition *interpret* organized sensor data. The distinguishing feature between the two classes is the abstraction from organized sensor data with information acquired from sensor data, for instance information about a monitored environmental phenomenon.

In discussing the design of a software system for publishing environmental observations Horsburgh et al. (2009) underscore the challenges of persistent storage and management, data access and communication, data interoperability, and data discovery. These challenges are typical to software systems that organize sensor data. Organization of sensor data can be achieved in various ways. Systems may build on conventional relational database management systems (Horsburgh et al., 2009; Junninen et al., 2009) or so-called "NoSQL" databases, such as Apache Cassandra. Systems may be tailored for streamed data processing (Bonnet et al., 2001; Carney et al., 2002; Madden and Franklin, 2002). Systems may use advanced data and knowledge representation languages. We highlight Semantic Web (Berners-Lee et al., 2001) technologies, which have found their application in sensor networks and sensor data (Sheth et al., 2008) with ontologies (Compton, 2011) and software architectures and systems (Moraru and Mladenić, 2012) being developed for the purpose of organizing sensor data.

Software systems that interpret sensor data build on organized sensor data and include computational techniques in, e.g., machine learning, inference, or complex event processing, to acquire information from sensor data. In this study, information is for *situations* and the acquisition of information is automated and may occur in (near) real time. Information is represented explicitly. For example, given organized observations for mean hourly concentration of particulate matter with diameter less than 2.5 μm ($PM_{2.5}$), complex event processing can be used to automatically and continuously detect situations of unhealthy exposure. The semantics of situations are, typically, different from the semantics of observations. Specifically to situations of unhealthy exposure, 'exposure' entails a longer time interval than mean hourly concentration and 'unhealthy' requires mean hourly concentration to continuously exceed a certain threshold.

Software systems that interpret sensor data are interesting for several reasons. First, the problem is harder than mere sensor data organization. The problem is known to various domains and several software architectures have been proposed in the literature (Clemente et al., 2013; Gorrepati et al., 2013; Conroy et al., 2011; Gaglio et al., 2007; Liu and Zhao, 2005; Whitehouse et al., 2006; Vassev and Hinchey, 2012). However, to the best of our knowledge, the work in this area is fragmented. Second, information acquired from sensor data is typically of more value to people than sensor data (Barnaghi et al., 2012). Of specific interest in this study are scientists and scientific computing on environmental sensor data. Third, for applications that build on large sensor networks and/or high frequency sensors it may not be desirable, or practicable, to persist sensor data for offline analysis. For such applications it may be best to acquire information over streams of sensor data, discard the sensor data, and only retain the acquired information.

The problems addressed in this study are (1) the heterogeneity of sensor data and (2) the explicit representation of situational knowledge automatically acquired from heterogeneous sensor data for environmental phenomena, specifically for SMEAR. Our aim is to use Wavellite (Stocker et al., submitted for publication) and demonstrate with a concrete application how it addresses these problems. Wavellite is a generic software framework aimed at the organization and interpretation of sensor data. It supports the processing of heterogeneous sensor data to sensor observations with homogeneous syntax and semantics; the mapping of sensor observations to dataset observations and the processing of datasets; the acquisition of situational knowledge from datasets; and the representation of situational knowledge. Extending our previous work (Stocker et al., 2013), the environmental phenomena of interest in this study are new particle formation and clouds. Hence, situations of interest are events of new particle formation and cloud events, occurring at Puijo. Information for such situations is acquired from data by sensors used for the monitoring of aerosols and weather at Puijo. To the best of our knowledge, Wavellite is unique in its support for the representation of situational knowledge acquired from heterogeneous sensor data for environmental phenomena.

The contribution of this work is two-fold. First, for readers interested in generic (and practical) approaches to the problem of representing situational knowledge acquired from sensor data, this work presents Wavellite and its application for a concrete use case in aerosol science, with real sensors and sensor data as well as using various computational methods, including machine learning. Second, for aerosol scientists and, more generally, scientists in domains in which sensors play an important role, this work presents a software system that integrates the processes of sensor data organization and sensor data interpretation. Specifically to aerosol scientists who study new particle formation, this work describes a software system that could support their workflows.

The paper is structured as follows. In Section 2 we provide a brief overview of Wavellite, specifically the logical structure of its architecture. In Section 3 we present the concrete implementation of the architecture. In Section 4 we briefly discuss how the implementation can be used in applications. In Section 5 we present our experiment on SMEAR sensor data and the representation of situational knowledge for events of new particle formation and cloud events. In Section 6 we discuss the results of our experiments and Wavellite more generally. In Section 7 we present related work. Finally, Section 8 draws some concluding remarks.

2. Architecture

We describe the logical structure of the Wavellite architecture to provide an overview of the layers, components, and modules as well as their responsibilities and interactions. The logical structure consists of four layers: measurement, observation, derivation, and situation. The four layers build on each other, from measurement to situation. Each layer serves a purpose and abstracts from underlying complexity. Fig. 1 provides a graphical overview of the architecture. Figure D.7 (Appendix D) gives an overview of the most important interfaces, in particular component interfaces with emit and execute operations and operation parameters.

Layers consists of components. Components are categorized in three broad classes: engine, reader, and writer. Components may execute information entities received on incoming streams and emit information entities to outgoing streams. Information entities are messages, specifically measurements and their contextual information, sensor observations, dataset observations, and situations. Components and streams form the nodes and edges, respectively, of a directed acyclic graph, known as a topology. Associated to components, the architecture includes modules. Modules are categorized in three broad classes: processing, learning, and store. Modules implement computations for purposes such as digital signal processing, machine learning, complex event processing, inference, retrieval and storage. The knowledge base is a third-party system.