# Demonstration: Real-Time Semantic Analysis of Sensor Streams

Harshal Patni, Cory Henson, Michael Cooney, Amit Sheth,
Krishnaprasad Thirunarayan

Kno.e.sis – Ohio Center of Excellence in Knowledge-enabled Computing
Department of Computer Science and Engineering, Wright State University
Dayton, OH 45435, USA
{harshal, cory, michael, amit, tkprasad}@knoesis.org

**Abstract.** The emergence of dynamic information sources – including sensor networks – has led to large streams of real-time data on the Web. Research studies suggest, these dynamic networks have created more data in the last three years than in the entire history of civilization, and this trend will only increase in the coming years [1]. With this coming data explosion, real-time analytics software must either adapt or die [2]. This paper focuses on the task of integrating and analyzing multiple heterogeneous streams of sensor data with the goal of creating meaningful abstractions, or features. These features are then temporally aggregated into feature streams. We will demonstrate an implemented framework, based on Semantic Web technologies, that creates feature-streams from sensor streams in real-time, and publishes these streams as Linked Data. The generation of feature streams can be accomplished in reasonable time and results in massive data reduction.

**Keywords:** Streaming Sensor Data, Abstraction, Semantic Web, Semantic Sensor Web

## 1 Introduction

Sensors produce huge amounts of low-level data about our environment that arrives in the form of rapid, continuous, and time-varying streams [3]. These data streams could quickly overwhelm any system not capable of effectively detecting and analyzing the most important data. Analyzing such sensor data streams and providing meaningful abstractions in real-time presents a significant research challenge. An abstraction, also called a feature, is a high-level representation of low-level sensor data.

There has been a lot of work in the database community on analyzing and mining real-time streaming data. Most of the current approaches within the database community provide mathematical summaries (i.e., minimum, maximum, average and count) for a single modality stream (like a temperature stream) over time (i.e. within a time window) [4]. These summaries are necessary and useful, but provide little help in answering questions involving real world events, such as: *Which weather stations are currently detecting a Blizzard?* Or: *What event (or sequence of events) is currently being detected by a weather station?*

The ability to answer such questions requires the semantic integration and inference over data from multiple single modality sensor streams using external domain knowledge. A feature-stream can be generated by aggregating a sequence of features detected by a particular sensor (or set of sensors), over a period of time. Feature-streams provide a clear and intuitive representation of how events evolve over time. An intuitive representation of trends in features will present decision makers with actionable situation awareness.

## 2 System Architecture

Consider the following question: *What weather events are currently being detected near Dayton James Cox Airport?* In order to answer this question, we would first need to find sensors near Dayton James Cox Airport, then access data streams for these sensors, integrate the streams capable of detecting the weather events, and finally, detect and represent the events.

The generation of feature streams requires a framework that can generate, integrate, and reason over multiple heterogeneous sensor streams. Reasoning over the integrated streams uses background knowledge and rules to generate feature-streams that represent events in the real world. The feature-stream framework is divided into four parts (see figure 1): (1) raw data generation, (2) data stream generation, (3) feature-stream generation, and (4) feature stream access.

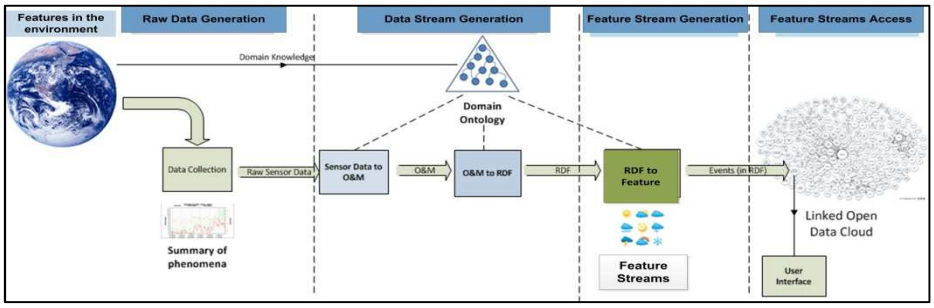


**Fig. 1. Framework for generating Feature Streams**

1 **Raw Data Generation:** The framework begins with the collection of raw streaming data from sensors within an environment. In this demonstration, we utilize MesoWest[1] , a project within the Department of Meteorology at the University of Utah, which provides near real-time access to weather sensor streams using a service API. Observations provided by MesoWest are encoded as CSV text, and includes measurements for temperature, visibility, precipitation, pressure, wind speed, humidity, etc. Example data provided by MesoWest can be seen below. The example contains information regarding the date and time of the observation, along with temperature (TMPF), wind speed (SKNT), and precipitation (PREC) observation values

```
PARMETER = MON, DAY, YEAR, HR, MIN, TMZN, TMPF, SKNT, PREC
VALUE = 11, 5, 2010, 13, 50, PDT, 30, 37, snow
```

2 **Data Stream Generation:** The second phase converts the stream of raw sensor data into an RDF stream. The raw sensor stream obtained from MesoWest is initially converted to Observation and Measurements (O&M)[2] format. O&M is a well-accepted XML standard in the sensors community. The SAX (Simple API for XML) parser[3] is used to generate the O&M XML stream. Below is an example encoding of the temperature, wind speed, and precipitation observations in O&M. The observation values for different time instants are separated using a block separator @@.

```
<swe:encoding>
```

[1] http://mesowest.utah.edu/
[2] http://www.opengeospatial.org/standards/om
[3] http://www.saxproject.org/

```
   <swe:TextBlock decimalSeparator="." tokenSeparator="," blockSeparator="@@"/>
</swe:encoding>
<swe:values>2010-5-11T13:50:00,30,37,snow@@</swe:values>
```

The O&M stream is then converted to an RDF[4] stream. RDF is a Semantic Web standard model for representation and interchange of data on the Web. XSLT[5] is used to convert the O&M to RDF, conformant to the W3C Semantic Sensor Network (SSN) ontology [6]. Below is an example RDF encoding of a temperature observation. [Note that *ssn*, *weather*, and *time* correspond to the prefixes for the SSN ontology, weather ontology, and OWL-Time ontology, respectively; *ssn-weather* corresponds to individuals generated by the system.]

```
ssn-weather:Observation_Temperature_KDAY_2005_10_21_5_30
   a ssn:Observation ;
   ssn:observedProperty weather:Temperature ;
   ssn:observedBy ssn-weather:System_KDAY ;
   ssn:observationResult ssn-weather:MeasureData_Temperature_KDAY_2010_05_11_13_50 ;
   ssn:observationSamplingTime ssn-weather:Instant_2010_05_11_13_50 .

ssn-weather:MeasureData_Temperature_KDAY_2010_05_11_13_50
   a ssn:SensorOutput ;
   ssn:hasValue "30.0" ;
   weather:uom weather:fahrenheit .

ssn-weather:Instant_2010_05_11_13_50_00
   a time:Instant ;
   time:inXSDDateTime "2010-05-11T13:50:00" .
```

3 **Feature Stream Generation:** The third phase integrates the RDF sensor streams and reasons over the integrated streams to detect features. Feature definitions are obtained from National Oceanic and Atmospheric Administration (NOAA)[6], and defined in the weather ontology. The feature definitions are initially used to filter the sensors capable of detecting a feature. A sensor is capable of detecting a feature if it is capable of observing all the phenomena that compose a feature. Filtering improves performance by reducing the number of sensor streams that are reasoned upon. SPARQL[7] is used for reasoning over the integrated sensor streams. An example SPARQL rule for detecting a Flurry over weather station KDAY is given below.

```
PREFIX ssn-weather:<http://knoesis.wright.edu/ssw/ont/ssn-weather.owll#>
PREFIX ssn:<http://http://purl.oclc.org/NET/ssnx/ssn/>
PREFIX weather:<http://knoesis.wright.edu/ssw/ont/weather.owl#>

ASK
{
   ?windSpeedObs ssn:observedBy ssn-weather:System_KDAY .
   ?windSpeedObs ssn:observedProperty weather:WindSpeed .
   ?windSpeedObs ssn:observationResult ?windSpeedResult .
   ?windSpeedResult ssn:hasValue ?windSpeedValue .

   ?snowObs ssn:observedBy ssn-weather:System_SB1 .
   ?snowObs ssn:observedProperty weather:Snowfall .
   ?snowObs ssn:observationResult ?snowResult .
   ?snowResult ssn:hasValue ?snowValue .

   FILTER(?windSpeedValue < 35)
   FILTER(?snowValue = "true")
}
```

The SPARQL rule is used to detect the most recent/current feature. A sequence of features detected over time results in a feature stream.

4 http://www.w3.org/RDF/
5 http://www.w3.org/TR/xslt
6 http://www.noaa.gov/
7 http://www.w3.org/TR/rdf-sparql-query/

4 **Feature Stream Access:** Finally, the feature stream is published as Linked Data [5]. The features can be accessed using either directly by issuing SPARQL queries to the RDF or through a map-based GUI[8].

## 3 Demonstration

During the workshop, a Google Maps based GUI will be demonstrated, showcasing the generated feature streams. The user can either select all the weather stations in a state, or search for a station by named location (using Geonames). Next the user is provided with an option to select features of interest. The system can currently detect blizzard, flurry, rain shower, and rain storm. Feature selection will result in the filtering of stations that are able to detect the features of interest. Clicking on a station shows the features detected over time along with the associated lower-level sensor observations. Because the features may not occur in real-time at the demonstration time, we will have a backup providing examples of interesting past events.

## 4 Evaluation

To evaluate the performance of this system, we collected 120 hours of data for sensors in (and around) Utah between February 2nd to 6th 2003. . Figure 2 shows an average of the amount of time (in ms) taken for each phase during feature generation. On average, for each hour, 427 sensors provided data during the evaluation, and produced an average of 1104 observations. 9 flurries, 1 rain shower, and 417 clear features were detected during the evaluation. We found an order of magnitude distinction between the number of observations and feature generated, which means storing only the features (if applicable) would result in massive data reduction. A demonstration page[9] will provide more details, including the storage evaluation

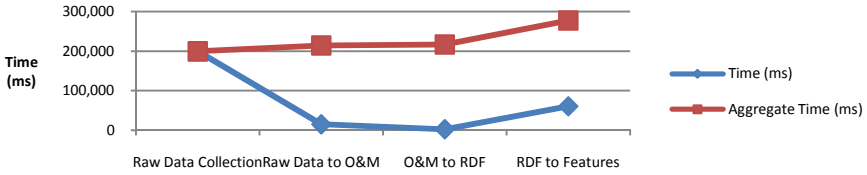


**Fig. 2. Performance Evaluation over Time**

## References


[1]. Gigaom Aricle on Big Data, 2010, http://gigaom.com/cloud/sensor-networks-top-social-networks-for-big-data-2/

[2]. Software must adapt or Die, 2010, http://www.readwriteweb.com/archives/data_analytics_software_must_adapt.php

[3]. Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom *Models and Issues in Data Stream Systems,* In Proceedings of the 21st ACM Symposium on Principles of Database Systems, 2002.

[4]. Refik Samet and Serhat Tural. 2010. *Web based real-time meteorological data analysis and mapping information system*. In Proceedings of *WSEAS Transactions of Information Science. and Applications,* September 2010, 1115-1125.

[5]. Bizer, C., Heath, T., and Berners-Lee, T. *Linked Data – The Story* So Far. International Journal on Semantic Web and Information Systems, 5(3), 1-22. Elsevier. 2009.

[6]. Lefort, L., Henson, C., Taylor, K., Barnaghi, P., Compton, M., Corcho, O., Garcia-Castro, R., Graybeal, J., Herzog, A., Janowicz, K., Neuhaus, H., Nikolov, A., and Page, K.: Semantic Sensor Network XG Final Report, W3C Incubator Group Report (2011). Available at http://www.w3.org/2005/Incubator/ssn/XGR-ssn/


[8] http://knoesis1.wright.edu/EventStreams

[9] http://wiki.knoesis.org/index.php/SSN_Demo