A Use Case in Semantic Modelling and Ranking for the Sensor Web

Liliana Cabral[1], Michael Compton[2], and Heiko Müller[1]

[1] Digital Productivity and Services Flagship, CSIRO, Hobart, Australia,
liliana.silvacabral@csiro.au, heiko.mueller@csiro.au
[2] Digital Productivity and Services Flagship, CSIRO, Canberra, Australia
michael.compton@csiro.au

Abstract. Agricultural decision support systems are an important application of real-time sensing and environmental monitoring. With the continuing increase in the number of sensors deployed, selecting sensors that are fit for purpose is a growing challenge. Ontologies that represent sensors and observations can form the basis for semantic sensor data infrastructures. Such ontologies may help to cope with the problems of sensor discovery, data integration, and re-use, but need to be used in conjunction with algorithms for sensor selection and ranking. This paper describes a method for selecting and ranking sensors based on the requirements of predictive models. It discusses a Viticulture use case that demonstrates the complexity of semantic modelling and reasoning for the automated ranking of sensors according to the requirements on environmental variables as input to predictive analytical models. The quality of the ranking is validated against the quality of outputs of a predictive model using different sensors.

Keywords: Semantic sensor data, Sensor ranking, Sensor Cloud Ontology, Viticulture, Predictive analytical models

1 Introduction

Real-time sensing for agricultural environmental monitoring has been an intense area of research (e. g., [6], [13]). Farmers and crop managers monitor crop growth and health continuously to make decisions based on their local knowledge and experience. Decision making is supported by environmental data as provided by automatic weather stations and by analytical tools that make predictions based on this data, e. g., predicting the risk of frost or plant disease.

In viticulture, for example, *botrytis bunch rot*, or *botrytis*, causes ripening bunches to rot on the vine [2]. Botrytis can develop during the ripening period and bunches may have to be harvested early at low sugar, having negative impact on wine quality. Wet weather is one of the factors that promotes the development of botrytis. To assist farmers, analytical models have been developed that simulate the effects of risk factors and control options on the development of botrytis epidemics. These models use weather information and crop management inputs to predict the risk that a major botrytis epidemic will occur at harvest.

The proliferation of sensing devices deployed in our environment benefits farmers in terms of enhanced situational awareness. There are, however, remaining challenges related to the deployment of sensors [5], the usability of decision support systems [16], as well as the discovery of sensors and re-use of data from different sensing devices. The latter is the focus of our work. Models that predict the risk of botrytis require specific sensor data as part of their input (e. g., air temperature, wind speed, and leaf wetness) and are sensitive to the quality of the input data. For a model to produce reliable results it is of great importance to run the model with input data that is fit for purpose.

Our contribution is the development of a generic method for analysing sensors based on their capabilities, observed phenomena, calibration history, previous measurements, and current state to find sensors that meet the requirements of a particular model. Ontologies that represent sensors and observations may help to cope with the problems of sensor discovery, data integration, and re-use, but need to be used in conjunction with algorithms for sensor selection and ranking. We present an algorithm that ranks available sensors based on their suitability as input for a given model. Model-specific requirements towards input data are expressed using a set of fitness functions. Our algorithm runs queries over the semantically annotated resources to evaluate these functions. The result is an ordered list of sensors that satisfy model requirements.

In addition, we discuss our experience in using ontologies to model knowledge about sensing data and sensing devices in the context described above. We build on the SSN Ontology [4] and ontologies derived from OGC's Semantic Web Enablement data models to provide an umbrella ontology, named Sensor Cloud Ontology (SCO), for our Sensor Web infrastructure. We discuss a number of use cases, in which we applied SCO.The design goal of SCO was to refine the semantic models provided by the imported ontologies according to the data and metadata captured by our infrastructure.

We evaluate our modelling and ranking based on requirements of a Viticulture use case that demonstrates the complexity of semantic modelling and reasoning for automated ranking of sensors according to the requirements on input data by predictive analytical models. The quality of the ranking is validated against the quality and sensitivity of the predictive model regarding different inputs.

The remainder of this paper is structured as follows. Section 2 describes our semantic sensor infrastructure and SCO. Section 3 gives details about application requirements through use cases. The semantic modelling to address the use cases in discussed in Section 4. Section 5 presents our ranking algorithm. We evaluate the results of our algorithm in Section 6. Section 7 concludes the paper by discussing related work and giving a brief outlook into future work.

2 The Semantic Sensor Cloud

The Commonwealth Scientific and Industrial Research Organisation (CSIRO) collects and archives data from a large number of terrestrial and marine sensors.

Parts of the data are made available via an in-house sensor data infrastructure, referred to as *Sensor Cloud*. An overview of the architecture is given in [14].

Within the Sensor Cloud, sensor data is structured following the hierarchy of $Network \rightarrow Platform \rightarrow Sensor \rightarrow Phenomenon \rightarrow Observation$. This structure resembles the deployment of sensors in the physical world, i. e., a sensor network consists of platforms, each platform has one or more sensors attached, and each sensor observes one or more phenomena. Besides observations, information such as sensor device characteristics and calibration history can also be accessed through the Sensor Cloud.

2.1 The Sensor Cloud Ontology

To semantically describe sensor data from the Sensor Cloud we created the *Sensor Cloud Ontology* (SCO)[3]. Figure 1 shows the main classes and properties of SCO. The principle behind its design is to use and extend existing ontologies, meanwhile aligning with the Sensor Cloud terminologies. Accordingly, classes and properties are created and mapped to the ones in existing ontologies. We reuse several ontologies, including the Semantic Sensor Network ontology (SSN)[4], the DOLCE ontology (DUL)[12], the OGC's Observation and Measurements (OM) ontology[5], and the GEO location (WGS84) vocabulary[6]. The advantages of reusing and extending existing ontologies are that sensor data can be queried according to the original terminologies while their consistency can be checked against SCO. We create SCO instances (RDF) from data in the Sensor Cloud using a system called SERAW, as described in [14].

SSN was designed to describe sensors: what is observed, how observations are made and the qualities of the sensors and observations [4]. SSN is built around a central pattern that describes the relationship between `ssn:Sensor`s, the `ssn:Property` measured, the real-world `ssn:Stimulus` that 'triggers' the sensor and the resultant `ssn:Observation` [9] . The ontology expands on this to describe the `ssn:MeasurementCapability` (`ssn:Accuracy`, `ssn:Frequency`, etc.) of the sensor as well as to provide a skeleton structure for describing `ssn:Platform`s and `ssn:Deployment`s.

Concepts that aggregate multiple sensors into larger units, such as sensor networks or sensors grouped by properties, are required by our applications. SSN provides `ssn:System` and `ssn:hasSubSystem` for describing multi-instrument units of technology. In SCO, such aggregations become `sco:Network`s, which can be used for wireless sensor networks, organisational networks, geographically related networks or mission oriented networks. These aggregations enable ranking and organisation of sensing resources: from `sco:Sensor` through `sco:Network` up to `sco:SensorCloud`. Such aggregations further enable the propagation of quality and trust estimates both up and down the hierarchy. For example, tending to increase the quality estimates of observations on a sensor because it is

[3] `http://www.sense-t.csiro.au/sensorcloud/ontology`
[4] `http://purl.oclc.org/NET/ssnx/ssn`
[5] `http://def.seegrid.csiro.au/isotc211/iso19156/2011/observation`
[6] `http://www.w3.org/2003/01/geo/wgs84_pos`

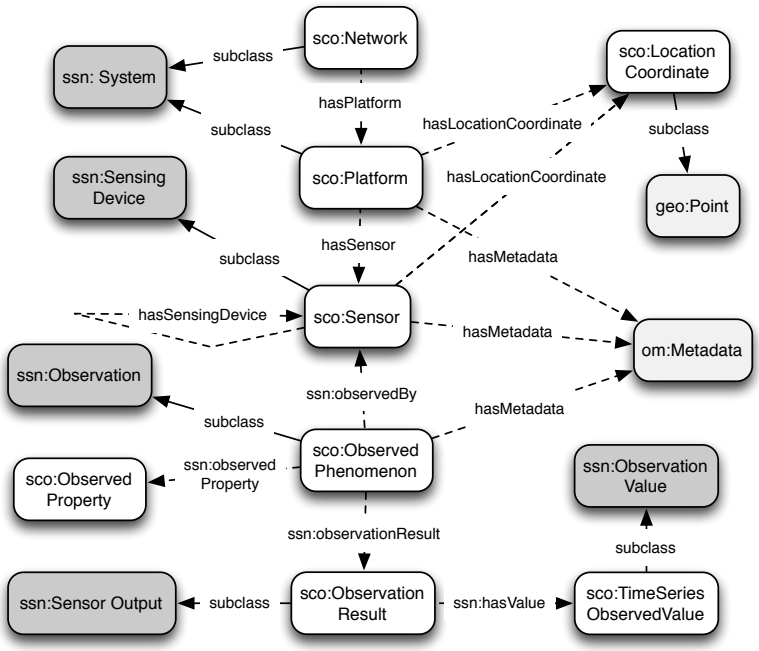

Fig. 1: Main Concepts in the Sensor Cloud Ontology

associated with networks of a known high-quality provider, or decreasing the trust on a provider because their sensors are not regularly maintained and calibrated.

To describe sensor observations in SCO, we introduced the four concepts `sco:ObservedPhenomenon`, `sco:ObservedProperty`, `sco:ObservationResult` and `sco:TimeSeriesObservedValue` as subclasses of concepts `ssn:Observation`, `ssn:Property`, `ssn:SensorOutput`, and `ssn:ObservationValue`, respectively. SSN leaves open the representation of time — fixing time only as a `dul:Region` — and does not include a definition of time series. We model time series as observation results of time-value pairs, by setting `sco:TimeSeriesObservedValue` $\sqsubseteq$ `sco:ObservationValue`, and with the `sco:TimeSeriesObservedValue`s having `sco:hasTimeValuePair` relations to `sco:TimeValuePair`s. In Section 4, we give more details of `sco:ObservedPhenomenon` and `sco:ObservedProperty`.

To be compatible with GEO and OM, we introduce `sco:LocationCoordinate` as a type of `geo:Point`, and use `om:Metadata` to describe metadata of several classes (e.g. `sco:Sensor`). In doing so, we are then able to use standardised (ISO) vocabularies for coordinates, deployment, and quality that are left open in SSN. Furthermore, we introduce some properties that are specific to the Sensor Cloud, e. g., those describing the number of time-value pairs of time series, and the first or last observation time.

3 A Use Case from Viticulture

Our work is informed by use case requirements in the viticulture domain. In this section we describe examples taken from the Sense-T project[7]. The project organized user workshops, from which requirements for environmental sensor data, derived variables and prediction models were collected from domain experts.

Predictive models for Botrytis risk use a combination of environmental variables. For example, the empirical *early-season* model in [2] assesses the effects of weather on botrytis risk. A botrytis risk index, called the *Bacchus Index* [10], uses air temperature and leaf wetness data to determine the daily risk of infection by Botrytis. The Bacchus Index is calculated for the given temperature at each "wet hour", which in turn is calculated from the leaf wetness duration. The hour is wet if the proportion of wetness duration is above, say 50%. Leaf wetness raw data measurements can vary for different sensors. As an example, the measurement can be the value of the electrical resistance on the surface of the sensor or, derived from that, the percentage of time that a leaf surface is considered wet[8].

The Accumulated Bacchus Index (ABI) is calculated from the daily sum of the Bacchus Index over a specified time period (e.g. from flowering to harvest). The early-season model plots the ABI each day and compares it against a threshold line, which is statistically determined from monitored botrytis epidemics over many years. This model runs in an ad-hoc manner. That is, at a point prior to the harvesting period the farmer decides to run the ABI. At this point, the challenge is to discover suitable sensors from a network of available sensors. We may find automatic weather stations or low-cost sensors deployed in a vineyard or in surrounding areas. In many cases suitable sensors may be deployed and operational, but if sensors are malfunctioning or not well maintained, data needs to be gathered from other suitable sensors. The choice of sensors can be based on their location as well as on their maintenance status and their current behaviour, for instance excluding sensors that have not been calibrated recently as well as sensors that show "unexpected" readings , for example, from interference to the sensing mechanism.

Although we have given the example of a predictive model, the same considerations apply to derived variables for the purpose of situational awareness. These include derived variables such as *Growing Degree Days*, *Evapotranspiration* or the *hourly Bacchus Index* (see for example [13]). From the user or application perspective, environmental variables derived from automatic weather stations sensor data streams are usually summarized in expressions like "Average daily temperature", "Total daily rainfall", "Maximum monthly discharge", "9am Relative Humidity", or "Average 10m wind speed (m/s)".

For the purpose of finding or reusing sensors, representing the meaning of these expressions can make the search or querying process much more efficient and precise. Say we are looking at measurements recorded as time series, these

[7] `http://www.sense-t.org.au/projects/viticulture`
[8] `http://www.campbellsci.com.au/237-l`

expressions constitute a combination of a functional (statistical) aggregation operator (e.g. average) and a temporal aggregator (e.g. daily) for an environmental property (e.g. temperature) with a unit of measure (e.g. degree Celsius). Composing properties with modifiers in a structured way can facilitate the reuse of sensor measurements that usually incur in very expensive computations. Usually these computations are necessary for the visualisation of time series, but they are equally important when used as input for predictive models.

Predictive models not only require observation data of a certain type, but also, in many cases, models have further (weak) constraints that define desired properties for the input data. One of the main constraints affecting the quality of the observation is the location of the sensors. For example, the sensors chosen for measuring temperature for the risk of frost in a vineyard should be the ones as close as possible to the frost event location. Another example of importance is proper sensor calibration. Data from poorly calibrated sensors can have a negative impact on forecast performance.

There might also be additional quality constraints that may require to examine values in time series e. g., no gaps longer than 6 hours, accuracy of all observations ≥ 0.95. In addition, we may want to propagate quality information along the hierarchy of `Network`, `Platform`, `Sensor`, and `ObservedPhenomenon`, for example a platform with many poorly calibrated sensors is considered low quality; if a network is considered poorly maintained the observations from the sensors are not reliable. Thus, in order to improve quality and reduce uncertainty in models, the discovery of sensors can involve finding information about sensor capability, deployment, calibration as well as measuring or ranking the quality of these factors so that they fit the requirements. Having knowledge of quality information for model input may also be used to estimate the uncertainty associated with the model output.

4 Semantic Modelling

In the following we describe the required semantic modelling and the solution given by the SCO and SSN ontologies according to the use case described in the last section. We look at environmental variables from two perspectives. First, as observed phenomena, where the observed property and the features of its measurement are represented. Then from the perspective of the sensor device and its influence on the quality of the measurement.

We use `sco:observedPhenomenon` to describe the context of the measurement for an environmental variable. This is aligned with the representation of `ssn:Observation` as a `dul:Situation`. As partially shown in the example below, this context includes a temporal modifier (e.g. daily), a functional modifier (e.g. total), the unit of measure (e.g. degree Celsius) and the sensor height (`sco` classes and properties accordingly). This is in addition to the inherited `ssn` class properties, which relate the observation to its observed property and result, among others. The `sco:ObservedProperty` is independent of an observation. It may be decomposed into parts using `sco:hasPropertyPart`,

which is a sub-property of `dul:hasPart`. This aligns with `dul:Quality` and thus sub concept `ssn:Property` being decomposable with `dul:hasPart`. We use `dul:UnitOfMeasure` as the super-class for units of measure from MUO[9]. Property `sco:hasUnitPart` can be used to decompose units of measure into parts.

```
<sco:ObservedPhenomenon rdf:ID="daily-sum-of-Bacchus-index">
    <sco:hasFunctionalModifer rdf:resource="#total"/>
    <sco:hasTemporalModifier rdf:resource="#daily"/>
    <ssn:observedProperty rdf:resource="#bacchus-index"/>
</sco:ObservedPhenomenon>

<sco:ObservedProperty rdf:ID="bacchus-index">
    <sco:hasPropertyPart rdf:resource="#leaf-wetness"/>
    <sco:hasPropertyPart rdf:resource="http://purl.oclc.org/NET/ssnx/
                         cf/cf-property#air_temperature"/>
</sco:ObservedProperty>
```

The decomposition of observed properties and units into smaller parts enables finding variables that are related to the derived variables in the user query. For example, a query for "daily temperature" can closely match any temperature that has a unit of measure equal to the "day" unit of measure or any of the "day"'s unit parts (e.g. minute, second). Similarly, a query for "degree days" can match related variables that are equal to "degree days's" property parts.

Regarding the quality of sensors, SSN has no capacity for describing calibration or maintenance, both of which often change a deployed sensor's properties (the properties described by `ssn:MeasurementCapability`). Calibration is adjusting a single device's output based on comparison against reference devices. Manufacturer specifications may describe the full accuracy range of a sensing device, but the actual performance of the device within this range depends on its particular calibration and the time and conditions since calibration. SSN does provide `ssn:Drift` as a way of describing the degradation of accuracy over time. But the interaction of calibration, time since calibration, expected drift, and dependence on calibration method, are subtle properties of a sensor that are important in some applications.

Our solution for this issue, as shown below, is to use `ssn:inCondition` to specify a time range for which the particular capability is valid. More specifically, we model `sco:CalibrationEvent` as a condition, in effect stating, for example, that the accuracy holds for the given calibration period. This approach is powerful because the combination of SSN capabilities and conditions can express many properties of sensors and the conditions under which they are true. Further, such a method is invariant to the passage of time and consequent changes in the world: to make a fixed specification, such as classifying a sensor as having accuracy 2.9% or classifying it into a concept such as `HighAccuracy`, means that any changes require retracting the assertion as well as any derived consequences and making new assertions. In our system, where changes in time affect our choices of the best sensors, and where we may need to revisit the history of a sensor (e.g. find the sensors that have always been well maintained), retraction and reassertion would be awkward and wouldn't give the required functionality.

```
<sco:Sensor rdf:ID="RIMCO_Rainfall">
```

[9] `http://purl.oclc.org/NET/muo`

```
 <ssn:hasMeasurementCapability rdf:resource="#calibration-capability-12-06-13"/>
 <ssn:hasMeasurementCapability rdf:resource="#calibration-capability-20-01-14"/>
 ...
</sco:Sensor>

<ssn:MeasurementCapability rdf:ID="calibration-capability-20-01-14">
  <ssn:inCondition rdf:resource="#calibrationEvent_20-01-14"/>
  <ssn:forProperty rdf:resource="http://purl.oclc.org/NET/ssnx/cf/
                                 cf-feature#rainfall"/>
  <ssn:hasMeasurementProperty rdf:resource="#RIMCO_Low_Rainfall_Accuracy"/>
</ssn:MeasurementCapability>

<ssn:Accuracy rdf:ID="RIMCO_Low_Rainfall_Accuracy">
  <sco:hasMaxValue rdf:resource="#max_low_rainfall_mm_s"/>
  <sco:hasMinValue rdf:resource="#min_low_rainfall_mm_s"/>
</ssn:Accuracy>

<sco:MaximumAmount rdf:ID="max_low_rainfall_mm_s">
  <dul:hasDataValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                                 >0.3</dul:hasDataValue>
  <dul:isClassifiedBy rdf:resource="#uom_mm_s"/>
</sco:MaximumAmount>

<sco:CalibrationEvent rdf:ID="calibrationEvent_20-01-14">
  <ssn:endTime rdf:resource="#calibration_date_20-01-14"/>
</sco:CalibrationEvent>

<dul:TimeInterval rdf:ID="calibration_date_20-01-14">
  <sco:hasIntervalDateTime rdf:datatype="http://www.w3.org/2001/
  XMLSchema#dateTime">2014-01-20T00:00:00</sco:hasIntervalDateTime>
</dul:TimeInterval>
```

Further to this issue, SSN gives no guidance on how to describe the measurement capabilities of an installed sensor instance versus the full range of potential properties defined for the device. For example, it is typical to describe properties of types of sensors using TBox assertions and properties of instances of such sensors as ABox assertions; if the TBox asserts that the device may have an accuracy of ±2–4%, and the ABox asserts that a particular instance of the device has accuracy ±2–2.9% (due to calibration), both are still asserted (using `ssn:hasMeasurementCapability`) for the instance because the TBox assertion is still inferred. Some method of distinguishing between general (possible) properties of sensors and actual properties of deployed and calibrated instances of such sensors is required. Our solution using instances (ABox assertions) according to SCO modelling is shown in the example above.

Attaching assertions, such as a sensor's accuracy or an assessment of its suitability as input to a model, to time points is in line with the fluents approach [18] and the modelling of the passage of time in DOLCE. The approach allows objects, such as sensors, to keep a fixed URI, but still show the variance of the object over time. As in the example below, it is easy to SPARQL for the latest accuracy assertion or all the quality assessments made on a sensor.

```
SELECT ?s (MAX(?time) AS ?lastCalibration)
 WHERE {
  ?s ssn:hasMeasurementCapability ?mc .
  ?mc ssn:inCondition ?c .
  ?c a sco:CalibrationEvent .
  ?c ssn:endTime ?ti .
  ?ti sco:hasIntervalDateTime ?time .
} GROUP BY ?s
```

5 Ranking of Sensors based on Fitness for Purpose

In this section we describe a ranking algorithm for selecting sensors that provide suitable input for a predictive model or to compute a derived variable. Our algorithm is based on the semantic modelling described in Section 4. It takes a description of model requirements for sensor data as input and returns a list of suitable sensors together with their observations. Results are ranked based on their fitness for purpose. In the following we describe the design of our ranking algorithm based on the definitions of fitness functions and fitness queries.

5.1 Fitness Functions and Fitness Queries

In our queries we distinguish *required properties* and *desired properties* for sensor data. Required properties (RPs) define data properties that are essential in order to run a model. A typical example is the observed property. If a model requires air temperature as input it does not make sense to use rainfall observations instead. We consider SCO classes and properties associated with `sco:observedPhenomenon` to define RPs, e. g., temporal modifier, functional modifier, and unit of measure. The temporal modifier defines the interval between observed values (e. g., hour). The functional modifier describes how observed values within the interval are aggregated (e. g., average). In addition, most models expect observations to be represented in a specific measurement unit (e. g., Fahrenheit).

Desired properties (DPs) describe preferences that models have towards the input data in order to ensure high quality results. A common example is sensor location. When running a predictive model we are interested in finding sensors that are in close proximity to the location for which the prediction is made (e. g., a vineyard). Other examples include calibration history (e. g., we prefer sensors that are calibrated regularly), age of the sensing device (e. g., newer devices are preferred over older ones), or state of the time series of observation (e. g., number of missing observations). In our ranking algorithm DPs are used to further restrict the set of sensors suitable to run a model as well as to rank results based on how they satisfy the DPs.

From our examples it becomes clear that there is a wide variety of DPs. To account for this variety we model DPs using *fitness functions*. We assume a set of fitness functions $F = \{f_1, \ldots, f_n\}$. Each function $f(G, op, KV)$ takes as input a knowledge base in the form of a RDF graph G, an observed phenomenon op, and a set of function-specific key-value pairs KV. Each function returns a value in an ordered domain D or a special null value $\perp$. A typical example is a spatial distance function. The function takes latitude and longitude information as additional parameters. It retrieves the latitude and longitude of the sensor that observed the given phenomenon and returns the Euclidean distance between that sensor and the given location. If latitude and longitude information does not exist for the given sensor the function returns $\perp$.

Fitness functions may also access external data sources. This is particularly important for functions that operate on time series. Currently we exclude the

actual observed time-value pairs from our knowledge base. The two main reasons for this decision are that (1) the data changes frequently and we would need to continuously update the knowledge base, and (2) RDF is not well suited to represent time series data potentially leading to increased query execution times. Observed time-value pairs can be stored using different architectures (e. g., the Sensor Cloud), instead. Our knowledge base contains a reference (URL) to the data. Using these references a fitness function can retrieve the time series from the external source and compute a value over the data (e. g., the maximal time gap between consecutive observations).

Formally, a query for sensors suitable to run a given model, referred to as *fitness query*, is a pair (Q, FFS). Q is a SPARQL query for observed phenomena that satisfy the RPs of a model. Note that we focus on observed phenomena instead of sensors since each sensor can observe multiple phenomena. We later retrieve the sensor that made the observations for our final result. *FFS* is a set of *fitness function statements*. Each fitness query contains a non-empty set of fitness functions $FFS = \{(f_1, \omega_1, c_1, KV_1), \ldots, (f_n, \omega_n, c_n, KV_n)\}$. With each function we associate a weight ω to reflect the importance given to that particular property. For example, a model may give higher importance to the distance of a sensor to a given location than to calibration status. Although fitness functions represent desired properties (or weak constraints), models can have thresholds on fitness values, e. g., only consider sensors that have been calibrated within the last 12 months. We associate a Boolean function c with each fitness function to represent such strong constraints. Function c returns *true* if the value returned by f satisfies the constraint and *false* otherwise.

5.2 Ranking Algorithm

Our ranking algorithm, which executes fitness queries, is shown in Figure 2. Information about sensors and their properties is maintained in a knowledge base in RDF format. The algorithm takes the RDF graph G, a SPARQL query Q, and fitness function statements *FFS* as input. It returns a ranked list of (sensor, observed phenomenon)-pairs. There are three main steps in our algorithm: (1) retrieve observed phenomena that satisfy the RPs, (2) compute a ranking of returned phenomena for each fitness function in *FFS*, and (3) compute an aggregate ranking of phenomena from the individual rankings.

The first step in the algorithm returns observed phenomena in G that satisfy the RPs by executing the SPARQL query Q. The only constraint towards Q is that it returns a list of URIs for `sco:ObservedPhenomenon`. An example query is shown below. The query retrieves all observed phenomena for the observed property `cf:air_temperature`, which have been measured hourly and classified according to http://purl.oclc.org/NET/muo/ucum/unit/time/hour.

```
SELECT DISTINCT ?observation
WHERE {
  ?observation a sco:ObservedPhenomenon .
  ?observation ssn:observedProperty cf:air_temperature .
  ?observation sco:hasTemporalModifier ?tempMod .
  ?tempMod dul:isClassifiedBy <http://purl.oclc.org/NET/muo/ucum/unit/time/hour>
}
```

Input: G, Q, FFS
Output: List of (sensor, observed phenomenon)-pairs and their overall ranking score

```
CANDIDATES ← SPARQL(G, Q);          /* Step 1 */
RANKINGS ← ∅;                       /* Step 2 */
for all (f, ω, c, KV) ∈ FFS do
    RANK ← ∅;
    for all op ∈ CANDIDATES do
        val ← f(G, op, KV);
        if c(val) then
            RANK ← RANK ∪ {(op, val)};
        else
            CANDIDATES ← CANDIDATES \ op;
        end if
    end for
    RANKINGS ← RANKINGS ∪ sort_f(RANK);
end for
RESULT ← ∅;                         /* Step 3 */
for all op ∈ CANDIDATES do
    score ← 0;
    for all (f, ω, c, KV) ∈ FFS do
        score ← score + (ω × get-rank-position(op, RANKINGS, f));
    end for
    RESULT ← RESULT ∪ {((op, get-sensor-for(op)), score)};
end for
sort_ASC(RESULT);
return RESULT;
```

Fig. 2: Pseudo-code for ranking algorithm.

The second step in the algorithm ranks candidate phenomena for each fitness function f individually. Should a candidate fail the respective strong constraint c associated with f it is removed from the set of candidates. For those candidates that satisfy the constraint we maintain the URI of the phenomenon and the returned function value. We then rank all candidates that have not been pruned. We assume the existence of a ranking function $sort_f$ that returns a list of URI-value pairs such that the pair with the best value (according to function f) is ranked first, the second-best value second, and so on. For a spatial distance function, for example, the pairs are sorted in ascending order of distance values.

The last step of the algorithm computes an aggregate ranking for candidates based on the individual rankings and the weights given to the respective fitness functions. We maintain a list of rankings ($RANKINGS$). The problem of combining ranking results has been studied extensively in social choice theory and web search (e. g., [7], [1]). Dwork et al. show that the problem of computing an optimal solution can by NP-hard (for certain distance measures) given four or more input rankings [7] and propose several heuristic algorithms for rank aggregation.

Our current implementation is based on Borda's positional method [3] that can be computed in linear time. Borda's method assigns a score to each candidate corresponding to the position at which it appears in each individual ranking. The function *get-rank-position*$(op, RANKINGS, f)$ returns the rank position of the candidate identified by op in the ranking computed for fitness function f. We multiply the result by the weight assigned to f. Function *get-sensor-for*(op) retrieves the URI of the sensor that observed phenomenon op. For our result we sort all (sensor, observed phenomenon)-pairs in ascending order of their accumulated score. Note that we can use other rank aggregation methods, e. g., using Markov chains as proposed in [7]. The main purpose of this section, however, is to present a method for discovering sensors that are fit for purpose based on the semantic modelling in Section 4. We consider the problem of evaluating the effectiveness of different ranking methods as future work.

Table 1 gives examples of fitness functions that we have implemented. The functions use SPARQL queries to retrieve information such as sensor location, most recent calibration event, or the URL that provides access to the time series of observed values. The spatial distance function expects coordinates for a point of interest (PoI). It computes the Euclidean distance between the PoI and the retrieved coordinates. The gap count function looks at the time intervals between consecutive values in a time series. If the interval between two values is larger than the maximum interval it increments the gap count by the quotient of the actual interval and the expected interval.

To give an example for fitness function statements consider a model like the accumulated Bacchus index that requires leaf wetness data. The DPs are (i) close proximity to a PoI, and (ii) calibrated recently. We use the following statements to express the DPs for location (f: Spatial Distance, ω: 0.8, $c(val)$: $val \leq$ `10 km`, KV: $\{lat : -42.15, lon : 147.45\}$) and calibration ($f$: Calibration, ω: 0.2, $c(val)$: $val \leq$ `6 months`, KV: $\{\}$). We exclude sensors that are more than 10 km away from the PoI or that have not been calibrated in the last 6 months. We also put higher importance on the distance of the leaf wetness sensor to PoI than on the calibration date, e. g., due to the model being more sensitive to location.

Table 1: Three examples of fitness functions for spatial distance, last calibration date, and number of gaps in a time series of observed values.

	Steps	Parameter
Spatial Distance	1) SPARQL for location of measuring sensor 2) Calculate Euclidean distance	Latitude for PoI Longitude for PoI
Calibration	1) SPARQL as shown on page 8	
Gap Count	1) SPARQL for URL to access data 2) Retrieve time series of observed values 3) Count number of gaps	Expected Time Interval Max. Time Interval

6 Implementation and Evaluation

We implemented a number of derived variables from our Viticulture use case. We used the Kepler [11] workflow engine to compute the variables, having the Sensor Cloud infrastructure as the source of raw data.

We use the Accumulated Bacchus Index (ABI) (see Section 3) to evaluate our modelling and fitness for purpose ranking algorithm. The ABI requires hourly air temperature and leaf wetness observations as input. Our evaluation consists of the following steps. We first compute a baseline ABI graph for a fixed period (Jan. 2014) using a pair of well maintained sensors for air temperature and leaf wetness, i. e., our gold standard. We then fix the leaf wetness sensor and re-run the ABI with different temperature sensors from a set of candidates. That is, we assume that we need to find a replacement for the gold standard temperature sensor. We plot the different ABI outputs in Figure 3. From these results we derive the optimal ranking of candidates based on how good they resemble the baseline result. We then show that our ranking algorithm produces the optimal ranking according to the user criteria.

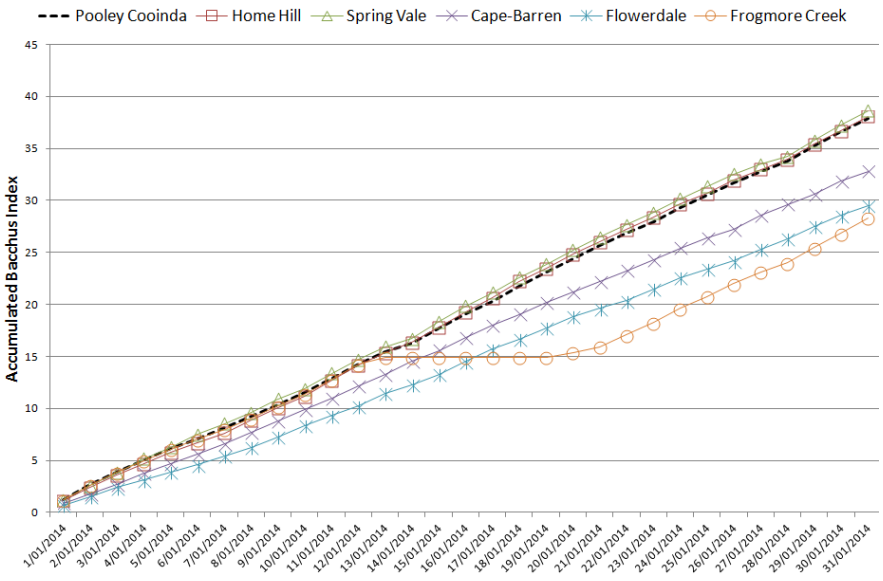

Fig. 3: ABI computed for Jan. 2014 using a fixed leaf wetness sensor (at *Pooley Cooinda*) and six different air temperature sensors.

Our baseline is obtained using sensors at *Pooley Cooinda*. Our set of candidates contains five air temperature sensors that are in close proximity to *Pooley Cooinda*. Figure 3 shows that the sensor at *Home Hill* results in an ABI that almost exactly resembles the baseline result. The sensor at *Spring Vale* gives

a less accurate but still acceptable result. Using the sensor at *Frogmore Creek* produces the worst result. This is mainly due to the fact that the sensor has a large gap (approximately seven days) with missing data. Based on Figure 3 we define an optimal ranking for candidates in order of increasing mean squared error (MSE) with the baseline (shown in brackets): *Home Hill* (0.008), *Spring Vale* (0.015), *Cape-Barren* (0.068), *Flowerdale* (0.110), *Frogmore Creek* (0.371).

We now show that our ranking algorithm can produce the optimal ranking based on user criteria expressed using fitness function statements. To do so, we use two different fitness functions: one for spatial distance and one for missing data (as shown in Section 5). For the latter we count the number of missing observed values considering that the ABI requires input on an hourly basis.

Table 2 shows rankings of candidate sensors using different fitness function statements. The ranking using the spatial distance fitness function solely uses distance from the leaf wetness sensor at *Pooley Cooinda* to rank the temperature sensors. According to this ranking the sensor at *Frogmore Creek* is the best replacement sensor and the sensor at *Cape-Barren* is the worst replacement sensor. The result clearly contradicts the optimal ranking. When ranked by increasing number of gaps, *Home Hill* is the best choice and *Frogmore Creek* the worst. This ranking exactly reflects the optimal ranking and highlights that the quality of data is of higher importance to the ABI than the distance of the sensor. The aggregated rankings are weighted over distance and gap count. When giving a higher importance to distance (Aggreg. (0.8, 0.2)), the sensor at *Frogmore Creek* is still ranked as one of the preferred replacements. The situation changes when giving higher importance to gap count instead.

Table 2: Rankings of candidate sensors for different fitness functions.

Rank	Spatial Distance	Gap Count	Aggreg. (0.8, 0.2)	Aggreg. (0.2,0.8)
1	Frogmore Creek	Home Hill	Home Hill	Home Hill
2	Home Hill	Spring Vale	Frogmore Creek	Spring Vale
3	Spring Vale	Cape-Barren	Spring Vale	Cape-Barren
4	Flowerdale	Flowerdale	Flowerdale	Flowerdale
5	Cape-Barren	Frogmore Creek	Cape-Barren	Frogmore Creek

We repeat the experiment for Apr. 2014 where none of the candidate sensors has gaps (graph not shown). The optimal ranking (based on MSE) is: *Frogmore Creek* (0.001), *Home Hill* (0.004), *Spring Vale* (0.011), *Flowerdale* (0.022), *Cape-Barren* (0.075). This ranking equals the ranking based on spatial distance (as shown in Figure 2). A ranking on gaps, on the other hand, will rank all sensors the same. Thus, for the different time periods different fitness functions produce the optimal result, i. e., gap count for Jan. 2014 and distance for Apr. 2014. For both time periods the aggregated ranking that gives higher importance to gaps always produces the optimal result. This clearly shows the benefit of combining fitness function statements to receive optimal results for different situations.

The evaluation shows that our algorithm is able to produce optimal rankings. The quality of ranking, however, depends on the user supplied fitness function statements. If the statements correctly capture the fact that the ABI has high sensitivity to gap count, our algorithm correctly ranks the replacement candidates. Should the user decide to give higher weight to the distance, on the other hand, the results could be non-optimal. In general, the choice of functions and their weights requires knowledge about the sensitivity of the model. We envision that developers of a model will specify these fitness requirements. The user then only specifies user dependent properties such as the location of their vineyard.

7 Conclusions

In this paper we propose an ontology-based framework for finding observation data that is fit for purpose as input to predictive models. In particular, we focus on the modelling methodology and approaches to handling the difficulties of describing sensors and evaluating them in a dynamic environment.

Within this framework, we further present a generic approach to ranking sensor data based on fitness for purpose. Our framework allows to rank sensors against criteria such as location, most recent calibration event, administered by an organisation that maintains sensors regularly, and data quality over a period of time (e. g., number of gaps). Ranking makes use of the accumulated metadata and semantics in the ontology for long term analysis and semantic analysis of the sensors. Since the quality of sensor measurements is not fixed over time our semantic model accounts for this changing behaviour.

Our work presents a first approach to exploit sensor properties for ranking based on fitness for a particular purpose. Existing approaches rank sensors based on the probability that they have a certain output state at a given time [8] or based on the similarity of their observed values [17]. The work in [15] goes into a similar direction as ours, however, our work has a strong focus on semantic modelling. Furthermore, the fitness functions that we consider allow for more complex expressions of fitness for purpose.

In future work, we consider automating the search for compatible sensors. For example, when a model requires maximum hourly air temperature, sensors that observe air temperature on a minutely basis are candidates because the data can convert through aggregation. Performance of evaluating fitness functions is another area of future work. Here, our particular focus is on caching the results of functions that incur high cost, e. g., computing values over time series data from external data sources.

Acknowledgements

This work was conducted as part of the Sense-T Program, and was jointly funded by the Australian Government, Tasmanian Government, the University of Tasmania, through the Tasmanian Institute of Agriculture, and the CSIRO Digital Productivity and Services Flagship. We would like to thank K. Evans, C. Peters, J. Forbes, D. Peel, S. Foster, D. Biggins and C. Sharman.

References

1. Ailon, N.: Aggregation of partial rankings, p-ratings and top-m lists. In: Proc. ACM-SIAM Symposium on Discrete Algorithms. pp. 415–424. SODA '07 (2007)
2. Beresford, R., Evans, K., Hill, G.: Botrytis decision support: online tools for predicting seasonal risk of botrytis bunch rot. Wine and Viticulture Journal 27, 46–52 (2012)
3. Borda, J.C.: Memoire sur les elections au scrutin. Histoire de l'Academie Royale des Sciences (1781)
4. Compton, M. et al.: The SSN ontology of the W3C semantic sensor network incubator group. Web Semantics: Science, Services and Agents on the World Wide Web 17, 25–32 (2012)
5. Corke, P., Wark, T., Jurdak, R., Hu, W., Valencia, P., Moore, D.: Environmental wireless sensor networks. Proceedings of the IEEE 98(11) (December 2010)
6. De Wolf, E.D., Isard, S.A.: Disease cycle approach to plant disease prediction. Annual Review of Phytopathology 45, 203–220 (2007)
7. Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web. In: Proceedings of the 10th International Conference on World Wide Web. pp. 613–622. WWW '01, ACM (2001)
8. Elahi, B.M., Romer, K., Ostermaier, B., Fahrmair, M., Kellerer, W.: Sensor ranking: A primitive for efficient content-based sensor search. In: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks. pp. 217–228. IPSN '09, IEEE Computer Society (2009)
9. Janowicz, K., Compton, M.: The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration into the Semantic Sensor Network Ontology. In: Proc. Int'l. Workshop on Semantic Sensor Networks. vol. 668. CEUR-WS (2010)
10. Kim, K., Beresford, R., Henshall, W.: Prediction of disease risk using site-specific estimates of weather variables. New Zealand Plant Protection 60, 128–132 (2007)
11. Ludäscher, B. et al.: Scientific workflow management and the kepler system: Research articles. Concurr. Comput. : Pract. Exper. 18(10), 1039–1065 (Aug 2006)
12. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A.: WonderWeb deliverable D18 ontology library (final). Tech. rep., IST Project 2001-33052 WonderWeb: Ontology Infrastructure for the Semantic Web (2003)
13. Matese, A., Gennaro, S.D., Zaldei, A., Genesio, L., Vaccari, F.: A wireless sensor network for precision viticulture: The NAV system. Computers and Electronics in Agriculture 69, 51–58 (2009)
14. Mueller, H., Cabral, L., Morshed, A., Shu, Y.: From RESTful to SPARQL: A Case Study on Generating Semantic Sensor Data. In: 6th International workshop on Semantic Sensor Networks in conjunction with ISWC 2013 (2013)
15. Perera, C., Zaslavsky, A., Christen, P., Compton, M., Georgakopoulos, D.: Context-aware sensor search, selection and ranking model for internet of things middleware. In: Intl. Conf. on Mobile Data Management. pp. 314–322. MDM '13, IEEE Computer Society (2013)
16. Shtienberg, D.: Will decision-support systems be widely used for the management of plant diseases? Annual Review of Phytopathology 51, 1–16 (2013)
17. Truong, C., Romer, K., Chen, K.: Sensor similarity search in the web of things. In: IEEE Symp. on a World of Wireless, Mobile and Multimedia Networks (WoWMoM). pp. 1–6 (2012)
18. Welty, C., Fikes, R.: A reusable ontology for fluents in OWL. In: Proc. Intl. Conf. on Formal Ontology in Information Systems (FOIS) (2006)