Department of Computing, Imperial College London

WikiSensing: A Collaborative Sensor Management System with Trust Assessment for Big Data

Dilshan Sumeshka Silva

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy of Imperial College of London

March 2014

Declaration

I hereby declare that this thesis entitled “WikiSensing: A Collaborative Sensor Management System with Trust Assessment for Big Data” is entirely my own work, except where specifically acknowledged in the text.

Dilshan Sumeshka Silva, March 2014

Copyright Declaration

‘The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work’

To the memory of Professor Moustafa Ghanem …

Abstract

Big Data for sensor networks and collaborative systems have become ever more important in the digital economy and is a focal point of technological interest while posing many noteworthy challenges. This research addresses some of the challenges in the areas of online collaboration and Big Data for sensor networks.

This research demonstrates WikiSensing (www.wikisensing.org), a high performance, heterogeneous, collaborative data cloud for managing and analysis of real-time sensor data. The system is based on the Big Data architecture with comprehensive functionalities for smart city sensor data integration and analysis. The system is fully functional and served as the main data management platform for the 2013 *UPLondon Hackathon*.

This system is unique as it introduced a novel methodology that incorporates online collaboration with sensor data. While there are other platforms available for sensor data management WikiSensing is one of the first platforms that enable online collaboration by providing services to store and query dynamic sensor information without any restriction of the type and format of sensor data.

An emerging challenge of collaborative sensor systems is modelling and assessing the trustworthiness of sensors and their measurements. This is with direct relevance to WikiSensing as an open collaborative sensor data management system. Thus if the trustworthiness of the sensor data can be accurately assessed, WikiSensing will be more than just a collaborative data management system for sensor but also a platform that provides information to the users on the validity of its data. Hence this research presents a new generic framework for capturing and analysing sensor trustworthiness considering the different forms of evidence

available to the user. It uses an extensible set of metrics that can represent such evidence and use Bayesian analysis to develop a trust classification model.

Based on this work there are several publications and others are at the final stage of submission. Further improvement is also planned to make the platform serve as a cloud service accessible to any online user to build up a community of collaborators for smart city research.

如果没有信任，我们不能袖手旁观

Translation: Without Trust, We Cannot Stand

- Confucius

"To be trusted is a greater compliment than being loved."
- George MacDonald

"What we need to do is learn to work in the system, by which I mean that everybody, every team, every platform, every division, every component is there not for individual competitive profit or recognition, but for contribution to the system as a whole on a win-win basis."

- W. Edwards Deming

Acknowledgements

First, I would like to express my sincere gratitude to my supervisor Professor Yike Guo, for being such a great advisor and introducing me to the field of sensor data management. I will always admire his enthusiasm and encouragement throughout these few years without which I would have not been able to reach such a successful conclusion to my thesis.

Second, I would like to profusely thank my second supervisor Professor Moustafa Ghanem for helping and guiding me through my research. His trust in my ability and patience has enabled me to gradually develop into a confident researcher.

Finally, I would like to thank Dr. Chandra De Silva, Dr. Anthony Rowe and Orestis Tsinalis as well as all my colleagues in the Discovery Sciences group at Imperial College, London for helping me on numerous occasions during this endeavour.

Contents

List of Figures

List of Tables

Publication List

- Silva, D., Ghanem M., & Guo, Y. (2014). *Managing Trustworthiness of Sensors in Collaborative Environments.* ACM Transactions on Sensor Networks (TOSN) (Under Review).

- Silva, D., Ghanem, M., & Guo, Y. (2012). *WikiSensing: an online collaborative approach for sensor data management. Sensors*, 12(10), 13295-13332.

- Lee, C. H., Birch, D., Wu, C., Silva, D., Tsinalis, O., Li, Y., & Guo, Y. (2013, October). *Building a generic platform for big sensor data application. In Big Data*, 2013 IEEE International Conference on (pp. 94-102). IEEE.

- Ma, Y., Guo, Y., Silva, D., Tsinalis, O., & Wu, C. (2013). *Elastic Information Management for Air Pollution Monitoring in Large-Scale M2M Sensor Networks*.International Journal of Distributed Sensor Networks, 2013.

- Guo, Y., Wu, C., Tsinalis, O., Silva, D., & Gann, D., *Wikisensing: Towards a cloud-based sensor informatics platform for life in a digital city.* Digital Futures, Aberdeen, UK, 2012: p. 23-25.

1. Introduction

Sensor devices are currently deployed almost everywhere for measurement and surveillance of various attributes of the environment [1]. A sensor can be defined as a device capable of capturing physical phenomena such as heat, light or motion about a physical system or an environment. Moreover these sensor devices can provide measurements of many properties such as pollution levels, temperature and road traffic. This research focuses on sensors that produce data streams that consist of a sequence of values (measurements and timestamps) or a recording of a measurement.

A sensor network is a collection of sensor nodes that collectively measure environmental changes. Sensor nodes take measurements and store them on-board or relay data towards remote systems [2]. With the growth of sensor networks, new technologies are required to systematically manage the streams of sensor data. Stream data is usually large, heterogeneous, real-time and continuous [3]. Moreover the use of online collaboration has largely proven to be an extremely powerful principle for sharing and gathering information [4] which can potentially be incorporated with sensor data. This helps to reduce the overall effort by combining the knowledge and experience of its collaborators. However such collaborative systems impose the important challenge of the need to assess the trustworthiness of the shared sensor data.

Clearly collaborative sensor data compares with the concept of Big data [5] a popular term used to describe massive volumes of structured and unstructured data that is far too complex for conventional databases to process. Moreover due to the similarities in the characteristics, collaborative sensor data can be considered as a type of Big data.

1.1. The Problem Statement

The increasing use and deployment of environmental sensors and wireless sensor networks [6] in different locations has recently given rise to the development and use of collaborative sensor management systems [7-11]. By using such systems, users can share both the collection and analysis of environmental data from different locations as well as build new applications that use such data.

The issue of managing sensor data is due to its large amounts, heterogeneous formats and continuous nature. So how can we provide a standard management system that can support large, continuous data with different formats? Another key fundamental challenge is how the data generated by sensors provided by third parties and not under our own control be trusted? Individual sensors could be faulty and reporting untrustworthy measurements for several reasons. They could have stopped working, be wrongly calibrated or beyond their life time. Sensors could even be hijacked by malicious attackers and forced to report wrong measurements. How would we be able to identify such situations, how would we define metrics to quantify trust and how should we reason about the trustworthiness of the sensors and their data?

Moreover if metrics are used how do we adapt these when the required information to calculate the metrics is unavailable or missing? Hence to summarise the aim of this research is to address the issues of sensor data management and to build a generic framework for managing trustworthiness of sensors in collaborative environments.

1.1.1. Motivation and Challenges in Sensor Data Management

Data management poses some important challenges when designing and developing a collaborative sensor data system with trustworthiness management. Such systems usually contain data from sensors, data due to collaboration as well as data on trustworthiness. The sensor data management issues are more conventional and are due to potentially large amounts of real-time data streams

from millions of different types of sensors deployed around the world. On the other hand collaborative and trustworthiness data management challenges are based on organising extensible amounts of related and unrelated data due to the absence of a generic and standard data representation methodology.

Providing efficient storage mechanisms are vital for most sensor data management systems as it needs to support large amounts of data. The storage strategy must be able to support heterogeneous data as well as the ability to support the querying of real-time stream data. The scalability of the data storage is also an important factor that needs to be considered, as the system potentially needs to handle growing numbers of sensor devices that send data continuously. Enabling users to annotate sensor data, sharing of information and managing extensible data are important factors that must be supported for effective online collaboration and trustworthiness management. The data management challenges discussed in this thesis are categorised as infrastructure, querying and information as well as the organisation and representation of information itself.

- *Infrastructure*: Designing a framework for scalable, efficient storage and retrieval of sensor data.
- *Querying*: The need to support querying of both real-time and historical information.
- *Information:* The need for aggregating data from multiple sensors as well as with data from reference sources.
- *Organising and representing information*: The challenge of representing extensible data as well as organising information provided by collaborating users.

1.1.2. Motivation and Challenges in Trustworthiness Management

To motivate this research, consider a simple scenario where a sensor owner registers a single sensor with a sensor data management system and makes the data available to other users. What are the attributes that can be used in order to trust

data from this sensor? One approach is to assign a reputation rating either, to the sensor owner (data provider), to individual sensors, or to the type of sensor (e.g. based on manufacturer information). These ratings could be provided by a trusted authority or crowd sourced by the users of the collaborative sensor management system.

Another approach could be based on gathering other information such as the historical readings from the sensor and analysing them to derive some trust rating based on past performance. This could be based, for example, on how many times previous sensor readings conflicted with our own background knowledge on what the measurements should be (e.g. if the previous readings are consistently outside normal ranges with no explanation). The background knowledge or ground truth can be information based on other research work (e.g. by the meteorological department, Universities, etc.) that provide data on minimum and maximum threshold measurements of locations. Hence normal measurements usually fall within the bound of such background data.

In addition, when there are multiple sensors deployed at or near the same location there is the possibility to compare the readings with another and identify if conflicts exist. In the absence of any contextual information that would justify why a conflict may reasonably occur, the existence of such conflicts in measurements could be an important indication that at least one of the sensors is not to be trusted. Moreover when assessing trustworthiness within a particular proximity, information on terrain, geographical locality, etc. can also be used. Such data can be considered as contextual data that impacts the trustworthiness of sensor measurements. For example, the communication of measurements from wireless sensors can be affected depending on the altitude of its deployment. In these cases the altitude may be used as contextual data to indicate its impact. However in practice there would be limitations in the availability of such data.

It must also be noted that the consequences may vary for incorrectly determining trustworthiness of sensor data. For instance, the consequences of false positives or false negatives on the trustworthiness of environmental sensor data

may not be too severe but can be quite significant when considering body sensor data (e.g. heart rate, respiratory rate, skin temperature, body posture, etc.)

To date, little work has been conducted in developing a generic trust modelling framework for collaborative sensor systems. Moreover, there is currently no standard, or agreed upon, definition for the concept of sensor data trustworthiness that can be used generically. There is also little work defining what information needs to be collected about the sensors, or their measurements, for use in a generic trust modelling framework. There are also no standard procedures to address the issue of missing information, for example, dealing with situations when necessary data is unavailable.

1.1.3. Motivation and Challenges of Collaborative Knowledge

One of the most important elements of open collaborative systems is the expert knowledge that is shared. This knowledge can be either information, data sets, ratings or annotations.

The effective use of such knowledge can help identify useful insights as well as aid in resolving certain problems. However this collaborative knowledge can sometimes be limited and also require certain transformations in order to be useful. For instance, we cannot expect all data in a collaborative environment to be rated or annotated by experts which is usually a gradual and time consuming process. Hence it can be challenging to make use of expert knowledge that may be incomplete or limited. Furthermore this data may lack structure that may lead to difficulties in converting this information to a standard format. Such standardisation is required so that the expert knowledge can be easily used for analysis or for any other types of data processing.

1.1.4. Challenges of Missing Data and Decision Making on a Multilevel

Trustworthiness management requires certain types of information to be available as explained in section 1.1.2. It is not practical in most situations to collect some of

these types of information (e.g. user views and ratings). What is the strategy that can be followed when information is partially or completely unavailable? One method would be to extrapolate the missing information from the available data and another method is to understand existing patterns of data in order to make estimations of the unavailable information.

The trustworthiness of sensors can be determined on information that is represented as a sequence of measurements (e.g. information based on temperature or pollution sensors) or represented as a multilevel. Sensor data that is represented as a multilevel can usually be further subdivided into smaller data items (e.g. measurements on route traces that are recorded by sensors can be further decomposed into smaller segments as discussed in chapter 8). When trust is assigned to information on a multilevel it is a challenge to compose the trust values of lower levels so that it is a correct reflection of the collective trustworthiness.

1.2. Summary of Contributions

The contributions of the research presented in this thesis are based on a collaborative sensor data management system for storing, querying as well as sharing sensor data and a trustworthiness management framework for assessing trust of sensor data.

The preliminary contribution of this research is a model for a collaborative sensor data management system. This model provides interfaces for connecting sensor devices and to enable online collaboration, a middleware for sensor data management and a storage model suitable for the efficient storage and retrieval of large volumes of data. The concept of virtual sensors is included in this model that enables the composition of sensor data streams into a single combined sensor, based on certain conditions. Moreover it supports the necessary constructs to manage real-time data and to aggregate data streams.

Based on this model the thesis introduces a system known as WikiSensing that incorporates collaboration with sensor data management. It is a publicly open system (wikisensing.org) with an API service layer for users to automatically connect sensor devices and query sensor data. WikiSensing follows a hybrid approach for data storage and uses a Wiki to enable collaboration.

This thesis also presents a new generic framework for trustworthiness management that is based on a generic probabilistic definition of trust. This framework can be used to capture and process sensor trustworthiness data. The trustworthiness of sensors is modelled using a set of attributes and metrics that are derived from the sensor data. Bayesian modelling is used to analyse these metrics and calculate trustworthiness ratings. This proposed model is evaluated using an air pollution monitoring scenario and a route data capturing scenario for the visually handicapped. The practicality and validity of the framework is also discussed based on these results.

The novelty of this research is based on providing a system that has a Wiki for supporting collaboration, a middleware API layer for service access, a hybrid data model for storage as well as a framework for providing trust assessment for sensor data.

1.3. Organisation of Thesis

The research work presented in this thesis is based on two main themes. The first concentrates on addressing the volume, velocity and variety challenges of Big data in the domain of collaborative sensor data management. The second focuses on the veracity challenge of Big data by addressing the trustworthiness of sensor data.

Chapter 1 This chapter introduces the challenges addressed by this work, a summary of contributions and an overview of the chapters of this thesis.

Chapter 2 A background study of other work that includes research on sensor data management and trustworthiness assessment as well as investigating proposed solutions on Big data challenges.

The following two chapters concentrate on the data management of sensor, collaborative and trustworthiness information.

Chapter 3 Describes the data management architectures of WikiSensing for managing sensor, collaborative and trustworthiness data.

Chapter 4 Contains the implementation details of WikiSensing and explains several key features of the framework. An evaluation based on the strategies of creating virtual sensors is also presented in this chapter.

The following three chapters explore the trustworthiness management of collaborative sensor data

Chapter 5 A standard probabilistic definition of trust, an extensible model and framework to assess the trustworthiness as well as an evaluation on the accuracy of the different Bayesian models used is presented in this section.

Chapter 6 Contains a set of case studies that demonstrate the integration of expert knowledge in WikiSensing.

Chapter 7 This chapter discusses several methods of extrapolating and estimating user ratings on sensor measurements and evaluates the trustworthiness management framework by incorporating such additional information.

Chapter 8 Describes the notion of a multilevel trust by extending the original trustworthiness model. The use of this model is illustrated using route data collected to aid the visually handicapped.

Chapter 9 The conclusions that are drawn and directions for future work is discussed in this chapter.

2. Big Data management for Sensors

The fundamentals of this research are based on the data management, online collaboration and the trustworthiness assessment of sensor data. Hence the background study is focused on discussing the details of other research work as well as existing solutions aimed at addressing the inherent challenges of these fundamentals.

Firstly it is important to understand the functioning of sensor networks and sensor data in order to identify their unique attributes. Moreover it is useful to characterise the different types of data management strategies and frameworks used for sensor data to recognise the various challenges associated with it. These various strategies and frameworks are classified into different generations of sensor data management in this chapter. This is an effective method to help understand the different challenges addressed as well as the novel features introduced. Once these challenges are realised these problems are then associated with the challenges of the increasingly popular notion of Big data [5]. Incorporating sensor data management with crowdsourcing via online collaboration and the use of data aggregation are also key features that are highlighted here.

The open nature of Big data, and the collaborative capabilities of sensor data management systems imposes another important issue of trustworthiness. Trust which is a generally considered a qualitative element needs to be represented quantitatively and moreover a standard definition is needed that can be used for sensor data. Hence other research work is reviewed to understand how they define, capture, represent, calculate and determine the trustworthiness of conventional as well as sensor data.

2.1. Sensors

The United States patents for sensor devices [12, 13] states that a sensor is a device that comprises of a substrate which is made of a metal, metallic oxide, semiconductor, dielectric or organic material. It can have projections or indentations formed on or in its surface with optional predetermined shape and dimensions. The structure of a sensor can undergo a chemical or physical interaction with the object to be detected (e.g. chemical sensor devices for detecting chemical amounts of gases, humidity, ions, etc., or physical sensor devices for detecting physical quantities of electromagnetic waves, temperature, etc.). Moreover a sensor is a device that measures a physical quantity and converts it into a signal which can be read or observed.

Sensor devices are generally small, low-powered, wired or wireless devices that are rapidly becoming cost effective to deploy in very large numbers. Sensors offer the ability to sense the environment densely, offering unprecedented opportunities for many scientific disciplines to observe the physical world [14].

2.1.1. Sensor Networks

Sensor networks [15, 16] provide infrastructure through which we obtain data about the physical, engineered, and social systems by using sensing devices. They have found a great deal of applications in the area of environmental monitoring, security surveillance, mental training, city planning and health care. In a sensor network, individual sensor nodes can be deployed in fixed locations or be on mobile devices, or can be ad-hoc nodes that connect or disconnect from the network. Each sensor collects measurements and exchange information through wired or wireless communication channels using various network topologies and communication protocols. In such a network the nodes of the network can be connected together or alternatively all nodes can communicate directly only with a base station.

Madden et al. [17] describes a system that is specifically designed for data acquisition and query processing in sensor networks. This work distinguishes sensor networks from other wireless, battery-powered devices; as they consist of tens or hundreds of autonomous nodes with limited battery power working collectively on remote environments to provide data.

2.1.2. Sensor Data

Sensor networks with a large number of sensors can produce great amounts of data that may be in various formats. A key characteristic of sensor data is that it is in the form of a stream that produces data continuously [18]. Due to this continuity the amount of data that a sensor or sensor network produce can be quite substantial. Moreover sensor data can be of various formats for instance, it can be in the form of a simple set of readings (e.g. temperature, humidity, pollution level, etc.) or can have a more complex or compound structure such as sensor devices producing measurements on *GPS* (*Global Positioning System*) route traces [19].

2.2. The Generations of Sensor Data management

This thesis makes no assumptions about the networking protocols used to connect the sensor nodes. It also makes no distinction between who owns or operates the individual sensor nodes. The main focus is on the data collected by the different sensors and made available for sharing and collaboration. Such data needs to be stored and managed in a system that enables users to collaborate.

Generation	*Features*	*Challenges Addressed*	*Examples*
First	Centralized or distributed systems, querying, aggregation and features	Storage and querying of sensor data, scalability, energy efficiency and real-time stream processing	*Aurora, The Cougar system, TinyDB*
Second	Limited collaboration by supporting the	Sharing information and aggregation and	*CitySense, The Discovery Net*

Generation	*Features*	*Challenges Addressed*	*Examples*
	configuration of sensor networks, processing and the development of analysis workflows on sensor data	analysis of data in sensor networks	*system, CitiSense*
Third	Collaboration on sensor data, trustworthiness management, processing of sensor data into virtual sensors	Big data challenges, collaboration and trustworthiness assessment across all sensor data	*Xively and WikiSensing*

Table 2.1: A summary of the generations of sensor data management systems

The categorisation is applied on different sensor data management systems with regards to supporting such collaboration into three generations as described below. Table 2.1 summarises the different generations of sensor data management with their distinct features and the specific challenges addressed.

2.2.1. The First Generation

It is quite natural that sensors produce a vast amount of data as they continuously monitor environments [3]. This was the design rationale for the first generation of sensor data management systems that focused on storing and querying the sensor data. Examples include *Aurora*, *Cougar* and *TinyDB* [17, 20, 21] which process incoming data streams for applications. Such systems provide query primitives and algebra containing several primitive operations for expressing queries over the streams and querying the sensor nodes in a distributed way. Such systems had no clear provisions for collaboration between users for the sensor data.

Aurora is a Database management system for managing data in monitoring applications developed by the *Universities of Brandeis, Brown* and *MIT*. This system processes incoming data streams by passing them through a data-flow system which then outputs a stream that can be used by applications. Queries

can be executed while the input tuples are run through this data-flow system. For instance, the filter operator that applies any number of predicates to each incoming stream and the aggregate operator that applies a function across a window of values in a stream. Once an input has worked its way through the paths of the flow it is generally drained from the system. *Aurora* can also maintain historical storage in order to support certain ad-hoc queries based on a persistence specification.

Developed by *Cornell University* the *Cougar* S*ystem* is a sensor data management system that supports querying in sensor networks. It follows a distributed query processing approach where the query workload determines the data that should be extracted from the sensors. The *Cougar System* uses an object-oriented database for storage and it models each sensor as a new Abstract Data Type (*ADT*). The stream processing functionalities are designed as *ADT* functions that return sensor data. It also supports long running queries formulated in *SQL* by extending the query execution engine by introducing a query construct known as '*every'*, specified with a *Time frame* parameter.

The sensor data management system of *TinyDB* specialises in query processing that uses acquisition techniques to reduce the power consumption of sensor devices. It first disseminates the queries to the sensor network and the query is then processed at the sensor nodes. Finally the results are collected back, up the routing tree that was formed as the query propagated. Hence it is clear that the intentions of sensor data management in this generation were to provide scalable and energy efficient storage systems that were able to handle large amounts of real-time data.

2.2.2. The Second Generation

The second generation data management systems provided certain primitives to support a limited amount of collaboration between users of sensor networks. These systems enabled either configuring the collection and/or the processing of data in a collaborative way between different users. For example, the *CitySense* [7] project

implemented and deployed an urban-scale wireless networking framework based on an open infrastructure allowing users to reprogram and monitor the same set of sensors via the internet and collect the data for shared analysis. The *Discovery Net system* [22] provides an example where different users could develop their own data collection workflows specifying how sensor data can be processed before storing in a centralized data warehouse. It also enabled them to develop analysis workflows for integrating the data with data collected from other data sources. Users of the system could thus share the same data and also derive new views and analysis results that were also shared.

The *CitiSense* [10] project is a distributed infrastructure to provide feedback on pollutants by the general public using mobile devices. By enabling this, the system supports enriching the information by the users and also allows them to comment on the operation and trustworthiness of the sensors. Each of the sensor management systems in this generation supports a degree of collaboration while operating on a fixed set of sensors. However, it is limited to either configuring sensors or sharing the processing of data of a specific sensor network.

2.2.3. The Third Generation

The third generation of sensor data management is based on open systems where users collaboratively submit data from any sensor and other users use this data. One example of this generation is *Xively* [9] (formally known as *Pachube* and then *Cosm*). It enables users to share their sensor data and allows collaborating users to build applications based on such data. The system however follows a passive approach with regards to the control (e.g. the ability to re-configure) of sensors by the collaborators when compared with some of the systems in the second generation. It simplifies online collaboration by allowing users to submit diverse data sets ranging from individual energy readings to data collected on various attributes of environments. Moreover, it allows developers to embed real-time graphs & widgets in websites; analyse and process historical data, and send real-time alerts to control devices.

Another third generation example is the WikiSensing System (wikisensing.org) [8] which is used in this research. It provides on-line database services allowing sensor owners to register and connect their devices to feed data into the system for storage. It also allows developers to connect to the database and build their own applications based on that data and perform different forms of analysis. It distinguishes from a system live *Xively* as it provides support for adding and annotating information about the sensors and their data through a wiki approach. Moreover it also supports the assessment of trustworthiness of sensor data.

2.3. Big Data Management

Big data is a common term used to describe the rapid growth and availability of large amounts of structured and unstructured data. Some of the current and popular examples of Big data are the data from the *Large Hadron Collider* (*LHC*) project, data from *Large Synoptic Survey Telescope* planned for northern Chile and data from the observation of events by sensors [23]. Big data management is the process of capturing, storing, querying and analysing these large and complex data collections. This section explores research work aimed towards addressing the main challenges of volume, variety, velocity and veracity in Big data. Volume and variety refers to the enormous amount of data that are provided by many sources with various structures. On the other hand the concept of velocity implies to the real time, continuous nature of the data. Moreover veracity in Big data refers to the ability to assess the reliability of such data.

Background on several other factors relating to Big data management is also discussed. These are based on aggregation and querying as well as collaboration and crowdsourcing.

2.3.1. Managing High Volumes of Data (Volume)

The multitude and wide spread distribution of sensors has generated a large amount of records or measurements [24]. These high volumes of data result in the issue of providing an infrastructure for data management that is scalable and efficient. For example, the infrastructure or framework must be capable of efficiently storing and retrieving large volumes of sensor information. It must also have the capacity to scale in order to handle large number of connected sensors that periodically submit data as well as a large number of users that concurrently access the system.

Google *Bigtables* [25] is a widely used (e.g. Google Finance, Google Earth, etc.) database to store large volumes of data in the range of petabytes. The data model of *Bigtables* is a set of processors known as clusters. Each cluster controls a set of tables. A table in *Bigtable* is a sparse, distributed, persistent multidimensional sorted map and the data is organized into three dimensions: rows, columns, and timestamps. Moreover *NoSQL* database (e.g. *MongoDB*, *HBase*, etc.) has become a popular solution for managing large volumes of data. This is due to the usually highly optimized key–value stores that are intended for simple retrieval and appending operations.

2.3.2. Managing Real-time Data (Velocity)

Sensors are frequently used to monitor the status of an environment continuously [26]. For example, a temperature sensor embedded in a fire-alarm system in a building continuously monitors abrupt changes in temperature or a wind speed tracking sensor and radar deployed in an aircraft to constantly detect and report the aircraft's location to a military system. Hence the continuous, real-time nature of sensor data has imposed several challenges when storing and querying this data.

The *Aurora* model [20, 21] proposed by *Daniel et al*. manages real time stream data for monitoring applications such as sensors that generate values at regular intervals. Their methodology is based on a data flow system that analyses the real-time data with special constructs introduced to support continuous

querying. *RAP* presented by *Lu et al.* [27] supports a real-time communication architecture for sensor networks. It provides a set of query/event *API* services based on a new real-time communication protocol. The service layer registers the query/event over an area over which sensors are usually deployed. The sensors then continuously send data to a base station or a central location. The communication protocol introduced by their work contains a set of efficient algorithms to support real-time querying. The *SPEED* protocol introduced by *He et al.* [28] also provides an *API* that supports end-to-end communication. This protocol maintains data communication speed across the sensor network by reducing end-to-end delays and providing congestion management for data packets.

On the other hand stream query engines such as *Esper* [29] and *SQLStream* [30] specifically provide high-level language constructs to query real-time stream data. *Esper* in particular provides open source components that can be integrated with programming platforms like Java and .Net, for application development. It provides a tailored Event Processing Language (*EPL*) based on event stream processing that enables expressing event conditions on large volumes of incoming messages or events. *EPL* allows registering queries such as obtaining an average value based on time or record windows in the engine. A listener class which is basically is then called by the engine when the *EPL* condition is matched as events (or measurements in case of sensor data) flow in. Similar to *Esper, SQLStream* is a processing platform for analysing and integrating high volume data streams that is however proprietary. Its *SQL* constructs supports time-series data processing with operators such as the window clause. Moreover with these queries executing continuously they process data as they arrive over row or time-based windows.

2.3.3. Managing Heterogeneous Data (Variety)

Usually a wide spectrum of data is available on the internet and found in various data sources with heterogeneous data formats. The heterogeneous data formats are in the nature of mismatches in the schema or data types. Similarly data provided by

sensors that are managed by different software can be heterogeneous in format [31]. Hence in order to support a wide range of data sources the data management needs to support the storage and querying of data with such discrepancies.

The importance of supporting a variety of data is identified by *Chamberlin et al.* [32] who presents an *XML* based query language for heterogeneous data sources. They use the versatility of *XML* to handle diverse formats of information to design this language. The work by [33] presents a design to simplify the specification of translations between a source and a target schema. Clearly there has been notable work to exploit the extensibility of *XML* [34] to provide a framework to manage heterogeneous data. However the drawback of *XML* is that there are yet no standard frameworks available to provide similar functionalities to those supported by conventional data management systems. Relational database systems [35] (e.g. *MySQL*, *Oracle*) have a solid data management framework but fail to support extensible data due to its fixed schemas. Non-relational databases [36, 37] on the other hand overcome this problem by alleviating the fixed schema constraints that enable the storage of heterogeneous data.

SStreaMWare by *Gurgen et al*. [31] is a service-oriented middleware for heterogeneous sensor data. It provides a global data schema to allow a generic data representation of various types of sensors. This enables declarative queries to be formulated according to this schema. Moreover *Aberer et al.*[38] also describe a middleware to manage the discrepancies in sensors by providing a layer of abstraction. Sensor data is represented using declarative specifications in *XML* and a specific controller interprets this information to obtain data from sensors.

2.3.4. Managing Data Trust (Veracity)

Due to the collective compilation of large amounts of data there is a possibility that it can contain uncertain or imprecise data. This is however in contrast to the traditional data warehousing approach where the data was always assumed to be

certain, clean, and precise. Big data circumvents this traditional architecture in order to accept enormous amounts of both structured and unstructured data at great velocity. By definition, unstructured data contains a significant amount of uncertain and imprecise data such as the data generated by sensors. Hence veracity is an indication of data integrity and the ability for users or an organization to trust such data.

The uncertain and imprecise data must be analysed in order to assess its trustworthiness. *Dai et al.* [39, 40] propose an approach to evaluate the trustworthiness of data from various sources based on data provenance. Their approach uses data provenance to include information on the process through which data has been generated. The data generation process is analysed to assess the trustworthiness of the data item, the data source and the data generated path. Trust is assessed for these elements based on similarities, conflicts and deduction of data. Data veracity is discussed more in detail in section 2.4, Trustworthiness Management.

2.3.5. Data Aggregation and Querying

Data Aggregation: It is the need for aggregating data from multiple sensors as well as data from reference data sources. These reference sources can be external data / sensor data providers such as the meteorological (e.g. www.metoffice.gov.uk) or transport (e.g., www.tlf.gov.uk) departments. Data gathering is a prerequisite for data aggregation, is the systematic collection of sensed data from multiple sensors into a centralised system of a single base station for processing [41]. Aggregation is required to combine data streams with each other to obtain combined readings and to combine data streams with reference data to obtain aggregated information. The challenge in aggregating different data streams arises due to the disparity of the sensor types, measurements, accuracy, quality of readings, time frames, etc. For example, the need to combine two temperature data streams that have different unit of measurements (e.g. Celsius and Fahrenheit) and are submitted in different frequencies with different time points.

Querying: Querying sensor data requires dealing with both real-time and historical information. Moreover the real-time nature and the continuous flow of sensor data have created the requirement for a near real-time processing of such data. Hence specific query constructs are required as this information arrives to the system continuously. The challenge arises when a query is processed and an output is produced, more up-to-date data arrive making the previous reading out-of-date. For instance assume that a query completes processing using a window of real-time data at the time frame t_1. This output will be invalid at time frame t_2 (where $t_2 > t_1$) as new data would have arrived. Also query constructs are required to mine historical information, when, for example, a user may wish to investigate sensor readings from a previous time.

2.3.6. Crowdsourcing and Collaboration

Crowdsourcing and collaboration is discussed by examining work on popular collaborative systems and the ubiquitous Wiki approach. Moreover current methodologies on collaborative sensors are also discussed here to demonstrate the importance and usage of combined sensor data.

Collaborative Sensors: The work in [42] presents a system with collaborating sensors using a sensor grid framework and a sensor grid client which is a collaborative session that enables meeting participants to share sensor information. These multiple collaborative sessions can interact with any combination of deployed sensors via this sensor grid. Collaborative sensor grids are a combination of sensor networks and grid computing. In this model each sensor gathers information from the environment and publishes it in real-time. A sensor adapter retrieves data from a connected sensor and communicates it to the sensor grid. The adapter provides among other capabilities a service interface to each sensor which facilitates the Grid integration and the Web service based management framework. This sensor adapter processes the raw sensor data and outputs the refined information.

The *QuakeSim* web service environment [43] integrates real-time and archival sensor data with high-performance computing applications for data mining earthquake data. This distributed computing infrastructure consists of Web services that provide access to data through well-defined programming interfaces (expressed in WSDL (www.w3.org/TR/wsdl)).

The research work by [44] describes virtual sensor networks based on collaborative wireless sensor networks. They define a collaborative virtual sensor network as a subset of sensors that collaborate to carry out a given application. These virtual sensor networks may exist simultaneously on a physical wireless sensor network, and the membership of the sensors may change over time. An area of this work is geographically overlapped applications. For example consider a set of sensors that are deployed to monitor rock slides and animal crossing within a mountainous terrain. The motivating factor is to have resource sharing where different types of devices that detect these phenomena rely on each other for data transfer without having to deploy separate networks. Similarly a goal of this research is to use the readings of existing sensors to obtain information where sensors are not currently deployed without the need of physically deploying them.

Wiki approach and rating methodologies: A wiki is a system whose users can add, modify, or delete its content via a web browser using a simplified mark-up language [45]. This approach has enabled quick access to information and the rapid production of data. Systems such as *Wikipedia* (*en.wikipedia.org*) and *WikiPathways* (*www.wikipathways.org*) are examples that successfully implemented the Wiki approach. Hence the Wiki approach provides the necessary infrastructure to obtain user annotations, feedback, etc. that leads to online collaborations.

The online wiki-like comment and self-moderation based systems of *StackOverflow* (Stackoverflow.com) and *BioStar* (biostar.stackexchange.com) specialises in answering domain specific questions. While *StackOverflow* focuses on computer programming-related problems, *BioStar* mainly concentrates on biology-based issues. These systems enable users to post their questions online

where experts are able to provide feedback by adding comments. These systems have proved to be a popular method of getting domain specialists around the world to comment and provide solutions to specific problems. As this is only a comment based system the user with the actual question has to manually distinguish the comments and use their own judgment in order to come to a certain conclusion. However the ratings on the comments or the answers through voting help the users to decide on the correctness of the response as well as to identify the trends of the collaborating users. These systems use the concept of tags which are keywords or labels that categorize a question with other, similar questions. This makes it easier for others to find and answer the questions. It keeps track of the unanswered questions in the system and ranks them in accordance with the number of users who viewed them. This drives the attention of users to answer these questions as the popularity and importance are highlighted. *StackOverflow* and *BioStar* use a rating system to assess the reputation of a user. This is based on the number of questions answered, edited posts and the scores rewarded.

Distinctions exist between these comment-based, wiki-like systems and a question answer sites such as *Yahoo! Answers* [46] or even conventional searching using Google. Firstly in contrast to *Yahoo! Answers* the information on the question as well as the posted comments that may amount to the answer can be edited as in a wiki-like fashion in *StackOverFlow* or *BioStar*. This enhances the collaborative power in dealing with specific problems as it ensures that the information in the questions and comments are more up to date. Secondly when compared with Google search, that can lead to outdated information on message boards as well as occasions when clicking through links may not actually get any results.

The work by [47, 48] discusses recommender systems also known as collaborative filtering. The importance of this methodology is that it uses previous information on user preferences to predict additional items that the user could potentially like. These are mostly popular with application associated with movies (*Netflix.com*), products (*amazon.com*), etc.

Online Collaborative systems: Online collaborative systems in the nature of the *Polymath Project* (polymathprojects.org) and the *OpenStreetMap* (www.openstreetmap.org) provide powerful infrastructures allowing people to obtain and share information. They have become the basis of knowledge sharing among users.

The *OpenStreetMap* project is a freely available map that covers the whole world and allows users to view, edit and manipulate geographical data in a collaborative manner [19]. It uses the knowledge on location information such as road, pathways and buildings provided by the users to build up comprehensive geographical maps. With over 320,000 contributors *OpenStreetMap* is a geographical source that provides data on maps without any technical restrictions on their use. It acquires data when the contributors provide location information using devices such as GPS, cameras and own observations. Similar to *Wikipedia*, *OpenStreetMap* enables any interested user to provide information.

OpenStreetMap has a set of rules for sharing knowledge that are based on simple logic which have proved to become extremely effective in the online collaboration process. For example data provided on a route within a short time period of a GPS signal is considered less accurate hence data received on routes taken from bicycle is deemed to have prominence over data received from a relatively faster moving car. *OpenStreetMap* obtains the knowledge from its users to annotate its maps, where the goal is to get the input of the users who are most familiar with these routes. Similarly in sensor data management a user who has knowledge about the local area would be more suitable to provide information on certain factors that would affect the reading of a particular sensor. For example, imagine a temperature sensor that is located in a building. A person who works or lives at that location would best know if there are certain factors affecting its reading such as a refrigerator or a heater. The knowledge of locals is a vital aspect as sensor devices can be located around the globe and there may be several factors influencing their measurements.

The Polymath project is an online comment based systems created to test if mass collaboration can be used to solve mathematical problems [49]. The method used to support the Polymath project was to use the commenting system in a blog and devise a series of rules [50] to govern how contributions should be made. The project was successful with over 40 people having contributed and resulted in at least two new publications. This concept continues to develop and has created over seven new Polymath projects to resolve various mathematical problems. Moderating and measuring of contributions, safeguarding the participants reputation and continual building of social connections that were based on the behaviour and psychology of participants are considered as the key aspects that lead to the success of this project. Users can contribute to the project by providing comments based on their knowledge and experience. The users collaborate with each other by these comments that create a discussion where ideas are instituted, exchanged and criticized.

2.4. Sensor Data Trustworthiness Management

One fundamental challenge facing the users of collaborative sensor data management systems is that there is little, or no, control on the quality of data collected by other users or on their validity. The key question here is whether users should trust the values reported in such data and use them in their own applications or not, and if so how do they assess the trustworthiness of such data values. This leads to other questions like whether such trustworthiness assessment simply depends on the user who submitted the data or depends also on the devices used to collect the data and the context in which these devices have been used.

2.4.1. Defining and Representing Trustworthiness

Trust in computer science is an extremely important factor when interaction occurs between computer systems or between humans and computers. A general definition of trust for internet based applications is described by [51] as the '*firm belief*' of the

competence of an entity to act dependably and reliably within a specific context. Trust can also be considered as a judgement when alternative sources of information are available [52]. Trustworthiness of data can be based on data integrity and data quality (the correct representation of data). One potential approach to assessing such trustworthiness is to start with an assumption that data is trustworthy only when more data items referring to the same real-world event have similar values [39, 40]. In such an approach, when conflicts occur, they then have a negative impact on the trustworthiness value and the provenance of data can be investigated by tracing its history of changes.

Trust can be determined based on policies or reputation [53]. Policies describe the required conditions or credentials that must be obtained for trusting a specific entity (e.g. sensor or data provider). Such reputation can be based on analysing and assessing the history of activities of the entity itself. Moreover, trust assessment can be based on a community view by considering how trust properties can propagate in a network of sensors and/or users [54]. Reputation is formulated using past sensor behaviours and this information is then used to predict future activities. This research uses sensors in a network to monitor the behaviours of other sensors to detect faults. Trust management systems such as *KeyNote* [55] and *PolicyMaker* [56] follow a unified approach to specifying and interpreting information needed to assess trust. These systems evaluate trust based on actions, principals, policies, credentials of distributed systems.

In [57] trust is considered as a subjective measurement of belief from one entity regarding the behaviour of another. This view exemplifies the idea that trust is a relationship between entities based on activities that relate to trust. The work by [58, 59] represents the trustworthiness in sensors as a probability that it corresponds to the actual measurement in the physical world. This work describes a methodology for assessing the trustworthiness of sensor data based on a subjective logic framework [60]. This methodology requires several intricate attributes such as the actual or forecasted sensor readings in order to provide a trustworthiness value.

Bayesian Networks [61] are a widely used method to represent and calculate probabilities for the management of uncertainty. Researchers at *NASA* describe a framework [62] that uses Bayesian Networks for sensor validation and diagnosis that is used as a guide to decide which sensors to trust. A Bayesian Network is used in this research to model the electrical power supply of an aerospace vehicle and is used for reasoning and querying sensor faults and inconsistencies.

Although some of these approaches address the issue of trustworthiness with specific solutions there is no standard or agreed method that can be used for sensor data in general. In this research we focus on providing solutions to assess the trustworthiness of available centralized sensor data. We do not concentrate on identifying the trustworthiness based on the relationship between the sensor and any intrusions or interferences as described by [63].

Once assessed representing this trustworthiness information can be a problem as it can contain extensible data as well as information that could be interpreted differently. Hence providing a common vocabulary is an important challenge in this research. Utilising ontology is a popular method in obtaining such common vocabulary. Several projects are discussed that were successful in capturing this type of information using different technologies.

SensorML (Sensor Modelling Language) [64] is a generic modelling language for representing the classes and relationships specific for sensors that can be instantiated to profile sensor devices. It provides a schema for persistent sensor data storage model for capturing sensor meta-data and sensor attributes. The *SUMO* (Suggested Upper Merged Ontology) ontology [65] is a top-level ontology for computer based information systems that provide concepts which are general throughout the knowledge domain. It is a single comprehensive ontology that was created by merging several publicly available ontologies. This is a good foundation for building more domain specific ontologies such as the *OntoSensor* ontology.

A more comprehensive ontology is the *OntoSensor* [66] and extension of *SUMO* that maps a subset of the *SensorML* concepts into *OWL* [67]. *OWL* has the capacity to formally describe the semantics of classes and properties used in Web documents. The work by [68] describes more specific ontologies such as the Sensor Hierarchy Ontology and the Sensor Data Ontology. They model information on sensor meta-data, physical properties and calibration details as well as concepts based on spatial and temporal observations and information on virtual transducers as a group of sensors that provides abstract measurements.

All these ontologies provide a rich set of concepts to represent a variety of sensor information and can be extended to represent the trustworthiness information of sensor to achieve a common vocabulary.

2.4.2. Metrics and Probabilistic Models for Trust

Software development use metrics to measure characteristics of processes in order to make improvements [69]. Metrics are a good way to provide a quantified measurement of a process under question. Usually measuring results with a single metric may not be sufficient; hence a set or a combination of metrics is used to measure the effectiveness of the process. The trustworthiness assessment of sensor data can be considered as a process that outputs a trust value which can be measured or quantified. The metrics can be generated to measure certain characteristics of the data that relate to trustworthiness during the assessment process. These metrics can then be used to identify the validity of the process, as well as the trustworthiness.

Probabilistic modelling has become a popular methodology for determining certain information based on other available data. These probabilistic approaches have proved useful to formulate and test hypotheses over quantities that are not exactly known. *Jordi and Sierra* [70] state that the main sources of information used by the trust and reputation models are based on experiences and information from third parties. Moreover *Krukow and Nielsen* [71] and *Despotovic*

and Aberer [72] discuss probabilistic models to determine trust in ubiquitous computing networks. These probabilistic models use the reputation based on previous behaviours and interactions of computer nodes. Trust in social media is emerging as a motivating and important challenge due to its growing popularity. The work by *Kuter and Golbeck* [73] proposes a methodology known as *SUNNY* to infer trust in social networks using probabilistic confidence models. It uses a probabilistic sampling technique to estimate the confidence in the trust information from designated sources. The uniqueness of this methodology is that it provides a confidence measure of the computed trust.

2.4.3. Trust in Collaborative Sensor Data Systems and Sensor Networks

Early collaborative sensor networks research includes *CitySense* [7] and *CitiSense* [10]. The former provides an open infrastructure for users to reprogram and monitor sensors via the internet. The latter, enables collecting feedback on sensor measurements on pollutants reported by the general public using mobile devices. The feedback is used for finding interesting patterns that can support decision making. A third system, *SenseWeb* [11] encourages sharing sensor information and for application development. These collaborative sensor systems all raise the issue of trustworthiness as a concern, due to the shared content provided by independent users that may have incorrect information. Although a partial solution is suggested by [11] in the form of community feedback based on ratings, however it is clear that these systems have the common problem of managing the trustworthiness due to their collaborative and open nature.

The trustworthiness framework proposed by *Generiwal et al*. [54] is specifically designed for sensor networks. This methodology exploits the advantages of sensors being able to pass information through the network. This enables the sensors in the network to maintain a reputation of other sensors in that network. Hence it must be noted that the goal of this research is to manage the trustworthiness of any sensor irrespective of it being in a sensor network.

Feature	*Other Available Systems*	*WikiSensing*
Storage	While the *Cougar System* and *TinyDB* provide a distributed storage, *Xively* stores sensor data centrally.	Centralised storage
Event processing for real time data streams	*Aurora* uses a data flow system, to process events. *Esper* and *SQLStream* provide high-level query construct for handing events.	Servers poll for up-to-date information and events form sensors.
Collaboration	Collaboration limited to a set of known sensors as well as known users in *CitySense*. *Xively* and *SenseWeb* provide shared infrastructure for the storage, management and usage of the data being collected.	Shared storage plus an online Wiki for sharing and annotating collaborative information on sensors.
Trustworthiness management	The framework by [54] determines trust by examining the reputations of sensors in a network by passing information.	Probabilistically determining trustworthiness of sensors in the system based on metrics.
Virtual sensors	In [44] virtual sensors are defined using a subset of sensors in a sensor networks.	Aggregate any compatible sensor streams to create virtual sensor based on proximity.
API Services for sensor data management	*Xively* provides a set of REST services.	Provides a set of REST services that also supports heterogeneous data formats
Heterogeneous Data	*SStreamWare* and the *Global Sensor Networks system* uses a middleware based on XML	Uses a non-relational database.
Commenting and Rating	*StackOverflow* and *BioStar* uses a Wiki approach	Follows a Wiki approach

Table 2.2: A comparison between surveyed systems and WikiSensing

Table 2.2 provides a comparison between the surveyed systems and the WikiSensing system. These features are discussed in detail in subsequent chapters.

2.5. Summary

This chapter discussed the background on sensor, Big data management, online collaboration and trustworthiness management exploring their characteristics. The different generations of sensor data management systems were also reviewed to understand the challenges they addressed. The next chapter presents the architectural design of the WikiSensing system highlighting the data management and the support for online collaboration.

3. Collaborative Data Management of WikiSensing

Some of the important challenges of designing collaborative data management systems for Big data are based on supporting manipulation and storage of various formats of high volume real-time data. Moreover the ability to assess the trustworthiness of this data is also highly sort-after. Designing a collaborative data management system for sensors impose similar challenges as this data possess comparable characteristics to that of Big data. Various sensor devices produce large amounts of measurements that are mostly heterogeneous and real-time. Furthermore the open collaborative features can introduce data from sources that are not reliable or trustworthy.

This chapter introduces WikiSensing, a collaborative sensor data management system with trustworthiness assessment. The architecture of WikiSensing is examined by describing its infrastructure for sensor data management, online collaboration and trustworthiness management. The data management challenges are founded on the Big data issues and are discussed initially. This is followed by detailed descriptions of the architecture on managing collaborative sensor data and a brief introduction to the architecture for trustworthiness assessment. A three tiered design strategy consisting of a middleware to service data requests between clients and databases is used in the WikiSensing system.

While the main focus of this chapter is to describe the architectures that address data management challenges, the next chapter discusses the implementation of these architectures demonstrating key functionalities of WikiSensing. This is followed by the chapters that focus on addressing the challenges of trustworthiness management of sensor data.

3.1. The Requirements (Challenges)

The data management challenges of designing a collaborative sensor system with trustworthiness assessment are due to the inherent characteristics of the data. This data includes information on sensors, collaboration and trust.

Firstly it is a challenge to manage sensor data as sensors can generate potentially large, real-time, heterogeneous measurements. Secondly as collaborative data contains different types of information (e.g. comments, annotations, ratings etc.) provided by various collaborators it is a challenge to organise, enable sharing and provide a common vocabulary for this information. Thirdly the extensible nature of trustworthiness information (e.g. trust metrics, contextual data, etc. discussed later in chapter 5) imposes issues of data representation. Expressing such data is a challenge due to the absence of a standard trustworthiness data representation methodology (e.g. ontology).

3.1.1. Managing Sensor Data

Managing sensor data is challenging due to the potentially large amounts of real-time, heterogeneous data provided by sensor devices deployed around the world. Providing efficient and scalable storage infrastructure for large volumes of data is essential for sensor data management. The infrastructure must also be flexible to store heterogeneous types of records as different sensor devices can produce measurements with different formats (e.g. single measurements such as the temperature or humidity or measurements with multiple dimensions such as distance, orientation and altitude). Furthermore the sensor data management infrastructure must support querying of real-time and historical data. In addition the ability to aggregate sensor data in order to produce useful information is another issue that must also be addressed. The data management challenges of sensor data are categorised as follows:

Infrastructure: Designing an infrastructure that is scalable, and provides efficient storage and retrieval of sensor information. The infrastructure must be capable of

efficiently storing and retrieving large volumes of heterogeneous sensor information. It must have the capacity to scale in order to handle a large number of connected sensors that periodically submit data as well as enable a large number of users to concurrently access the system.

Querying: The framework needs to support the manipulation of both real-time and historical information. Querying constructs are required to capture information that arrive at the system continuously. The real-time nature and the continuous flow of sensor data have created the requirement for a near real-time processing of such data. The challenge arises in case, where a query is processed and an output is produced, more up-to-date data arrive making the previous reading out-of-date. For instance assume that a query completes processing using a window of real-time data at the time frame t_1. This output will be invalid at time frame t_2 (where $t_2 > t_1$) as new data arrives. Moreover query constructs are also required to mine historical information, when, for example, a user may wish to investigate sensor readings from a previous time frame.

Information: The framework needs to support the aggregation of data streams from multiple sensors as well as information with reference data sources. The reference sources for example can be data providers such as the meteorological (*www.metoffice.gov.uk*) or transport (*www.tlf.gov.uk*) departments. A data stream is the term that is used throughout this thesis that refers to a collection of measurements transmitted by a sensor. Aggregation is required to combine these data streams with each other to obtain composite sensor measurements as well as to combine data streams with reference data to obtain aggregated information. The challenge in aggregating different data streams arises due to the disparity of sensor types, measurements, accuracy, quality of readings and time frames. For instance, consider the combination of two temperature data streams that have different unit of measurements (e.g. Celsius, Fahrenheit, etc.) and are submitted at different frequencies and hence have different time points.

3.1.2. Managing Collaborative data

The challenges of managing collaborative data are related to the organisation and the sharing of information that are provided by collaborating users. These issues are based on organising as well as providing a common vocabulary for the collaborative data. The collaborative data management challenges are categorised as follows:

Organisation of information: This is based on the challenge of organising sensor data and information provided by collaborating users. The collaborative information can contain data on the sensor environment (e.g. deployment information, comments on factors that impacts the trustworthiness of sensors, etc.), or on the sensor meta-information (e.g. accuracy, range, etc.). It can also be on sensor measurements (e.g. ratings on trustworthiness of measurement, annotations justifying anomalous measurements, etc.) or about any contextual details (e.g. measurement impacting factors such as factories or hospitals, details on sensor calibration, etc.). It is a challenge to organise this information as different users have diverse goals, views and can provide different types of annotations and this cannot be accommodated in a fixed schema. Moreover the need to associate and reference different types of information is needed when organising information to enable effective collaboration.

A need for a common vocabulary: Even when the collaborative sensor data is organised it is still a challenge to provide a common vocabulary [74] in order to preserve the correct semantics of the information. Certain terminology can have the same meaning for instance; with different users annotating sensors there is a broad chance of the existence of disparate terminologies that share common semantics.

3.1.3. Managing Trustworthiness Data

It is a challenge to manage trustworthiness data as it can be extensible as well as requires a logical representation of relationships. This data is extensible as new trust metrics and contextual data can be added when needed. Moreover the

relationships between this information need to be represented to demonstrate a natural classification (trustworthiness model described in chapter 5). Additionally trust metrics can also be assigned at multiple levels (discussed in chapter 8) and requires a hierarchical representation.

Representation of extensible and multilevel data: The flexibility offered by the framework to manage extensible as well as multiple levels of trust metrics poses the challenge of representation. A sensor or a sensor measurement can have several trust metrics or contextual data (relating to the trustworthiness) associated with it, further in certain scenarios these trust metrics can be classified into a hierarchy of sub levels. Moreover these metrics can also be based on measurement window sizes or parameters that influence its calculations. Hence there is a need to capture the trustworthiness information so that it correctly represents the circumstance of the sensor (or sensor measurement), the state of the calculations as well as the relationships between the metrics.

3.2. Infrastructure for Sensor and Collaborative Data Management

The infrastructure of WikiSensing for managing sensor and collaborative data has a layered architecture with a database, an application and a client layer. Figure 3.1 illustrates this architecture that includes the main components responsible for the management of sensor data and collaboration. This architecture addresses some of the fundamental Big data challenges that were already discussed (section 3.1) based on the efficient management of large volumes of data, the storage of heterogeneous types of records and the capturing of real-time information.

The collaborative sensor data management of WikiSensing has a client web interface as well as a set of *API* web services to connect sensors and manage their data. The collaboration features of the system are enabled using a *MediaWiki (www.mediawiki.org/wiki/MediaWiki)* that is tied with the WikiSensing web interface used for sensor data management. Recording annotations, comments and

rating on sensor data are the main functions of this Wiki. The Ontology data in WikiSensing is maintained by a third party *Virtuoso* (*virtuoso.openlinksw.com*) Ontology repository that manages *RDF* (Resource Data Framework) and *XML* data.

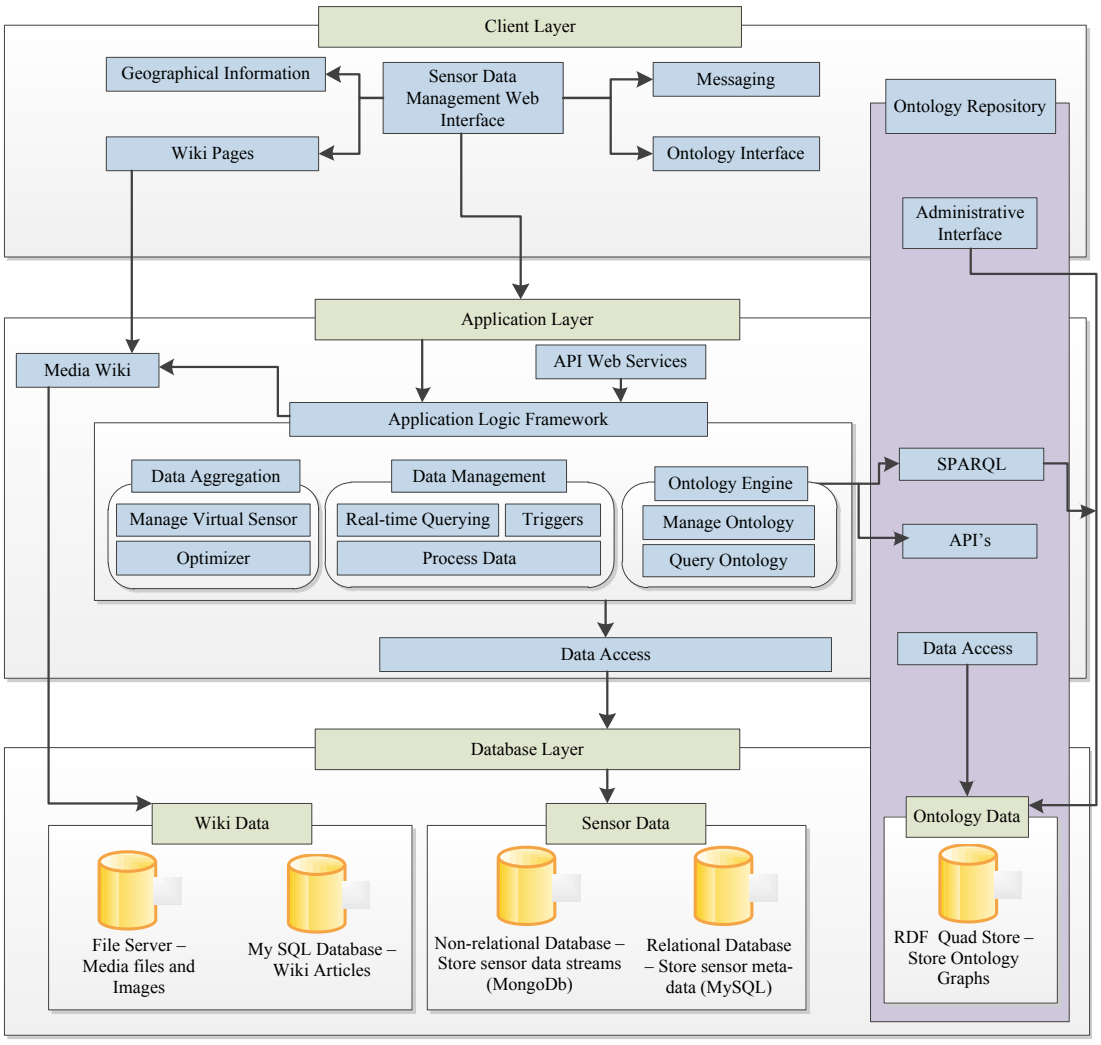


Figure 3.1: The Architecture of WikiSensing

The three tier architecture strategy of the WikiSensing data management framework enables the logical separation of functionality of presentation, application processing, and data management. This encompasses the flexibility for other applications to reuse functionality of each layer. Moreover it is easier to

distribute the layers over multiple physical tiers which can make good impact on application by improving scalability and maintenance. In addition it enables teams to work on different parts of the system parallel with minimal dependencies as well as test components independently of each other.

The database layer follows a hybrid model that consists of relational, non-relational and ontology storage strategy to address the heterogeneity of data. Special query constructs are used to support real-time data to obtain most up to date sensor measurements. The direction of the arrows illustrates the flow of control. The database tier hosts the databases for the sensor and the wiki data. There is also an ontology repository to store and manage the sensor ontologies. The application layer directly interacts with the database layer via the data access module. The application tier contains the modules for the *REST API* services, the application logic framework, controlling user access, supporting online collaboration and access to sensor ontologies. The client layer contains a web interface that has the capabilities of sensor data management, a Wiki front-end to support online collaboration, a graphical module for displaying geographic data, a messaging module for event handling and an ontology module for managing ontologies.

3.2.1. The Client Layer

The front end of the system is a graphical user interface consisting of a series of web pages and Wiki pages. The main web interface is implemented using *C# ASP.NET 3.5* [75] technologies and the Wiki pages are provided using a *MediaWiki* that resides in the application layer.

The advantage of using *ASP.NET* is that it can dramatically reduce the amount of code needed to build large applications [76]. *ASP.NET* framework is also complemented by Visual Studio integrated development environment designer tool that enables drag and drop controls, firewall and automatic deployment. Moreover all processes are carefully controlled and managed by *ASP.NET*, so that

in case a particular process is dead, a new process can be created in its place, which helps keep the WikiSensing web server continuously available to requests. On the other hand the use of *MediaWiki* software was rationalised due to it supporting the seamless management of collaborative information. For example, *MediaWiki* software is used by the English Wikipedia, the largest wiki in the world, with more than 4 million pages, 600 million edits since the project's inception [77].

The geographic information component can be an external system such as *Google maps API (code.google.com/apis/maps/index.html)* [78, 79] that enable the users to incorporate location information. The messaging component in the client layer is implemented using *PostBin (www.postbin.org)* that allows users to register certain *URLs* so that asynchronous requests can be logged when events occur. Moreover the *PostBin* is exclusively used in WikiSensing for sending messages in the case of triggers. A third party *Virtuoso* administrative interface is also available at the client layer to manipulate the ontology data. This facility is needed to maintain and update existing ontologies by authorised users.

3.2.2. The Application Layer

The application layer or middleware of WikiSensing comprises several components that are collectively responsible for the control and the management of the data and the users. These components contain the rules and the algorithms that are needed for sensor data management. The WikiSensing middleware is based on the *ASP .NET* framework that implements a model-view-controller [80] software design. The main advantage of using such a framework is based on its clean separation of functionality [81] that is required for implementing WikiSensing's layered architecture. The *ASP .NET* framework supports user management for the web application and *API* web services. The functionality includes validating user credentials, creating and modifying membership users, and managing user settings such as passwords and e-mail addresses.

The *Application Logic Framework* component contains the code and algorithms needed to co-ordinate and control other components. Moreover it includes the operations to invoke the functionalities of the other modules within the application layer. For instance, it executes the functionality to register a sensor device in the *Data Management* component and creates a corresponding Wiki page using the *MediaWiki*. The *MediaWiki* component that hosts the sensor Wiki runs on a *PHP* (*www.php.net*) framework on the application server. The *PHP* framework implements the security policies and rules that are prescribed by the *MediaWiki* for the user management of the wiki users in order to control access to its information.

The *Data Management* module supports historical and real-time querying, setting up triggers on data streams and processing of data. The framework supports queries that select sensor details (e.g. sensor readings, deployment information, etc.) as well as aggregate queries that combine several data streams. The architecture also makes provisions for continuous, real-time querying that provide data to users uninterruptedly within a specified period of time. These queries are managed by a separate server that polls for up-to-date information from the specified sensor. This server accepts query requests and replies back to the client either when new data arrives or continuously based on a time window. The processing of queries however does not consider the rate of data arrival and it is assumed that the server is able to manage the data frequency.

The *Triggers* sub-component is mainly used to inform users when certain thresholds are reached on sensor data streams. This is particularly useful to provide alerts in the case of abnormal or unusual sensor measurements. The *Data Management* module contains the *Process Data* sub-component to validate and process the data that is submitted and returned from the system. For instance certain data that is stored in the system is checked using this functionality (e.g. maintaining maximum and minimum measurements of a sensor, validating input data, etc.).

The *Data Aggregation* module provides the functionalities to create

virtual sensors by combining data streams. Moreover it optimizes querying to enhance the performance of the aggregation of virtual sensors. The *Optimizer* module focuses on increasing the efficiency of the aggregate queries which are considered as one of the most common operations in sensor data management [24]. It is responsible for analysing the information that contains the data streams that constitute the virtual sensors and identifying the most efficient (with minimal amount of database reads) methodology for aggregation. This also controls the storage of the virtual sensor readings in a cache repository for quick access. The *Data Aggregation* module contains synchronisation mechanisms needed to resolve discrepancies of time frequencies when aggregating multiple sensor streams. Example strategies on addressing such discrepancies are discussed in section 4.4.2 page 81.

The *Ontology Engine* module is the application logic interface with the *Virtuoso* ontology server. This enables the *Ontology Interface* at the client layer to query and update ontologies. The *Virtuoso API* supports raw *SPARQL* queries as well as specified querying functionalities that can be used to query the ontological data.

The *API Web Services* exposes the functionalities of WikiSensing in order to be used from different programming platforms. These services access the data from the underlining database server via the business logic imposed by the data management module. The *Data Access* component contains the operations for reading and writing the data to the database layer. The *Data Management* and *Data Aggregation* components access the databases using this module.

3.2.3. The Database Layer

The database layer of WikiSensing contains centralised databases for sensor data, wiki data and the ontological data. Each database is designed to run on separate servers with multiple server instances for each non-relational database cluster containing the sensor measurement streams.

The sensor data is stored using a hybrid database strategy with sensor deployment and configuration information stored in a relational *MySQL* [82] database, the sensor measurements stored in a *MongoDB* [36, 83] non-relational database and sensor properties stored using an ontology. The wiki data for the sensor information controlled by the *MediaWiki* software is stored in a *MySQL* database and any media files including images and videos are stored separately in a file server for efficient access. The textual information of the wiki pages is stored in multiple languages in a database server. The sensor ontological data is stored as ontological graphs in the repository using a *RDF quad store* [84]. This ontological database is under the control of the *Virtuoso* ontology repository with access points to WikiSensing at the application layer. The design strategies of the databases for the sensor data are discussed in detail in the following section. Table 3.1 summarises the storage strategies used by WikiSensing, moreover the trustworthiness information is also included here for completeness and is discussed later in detail in chapter 5.

Data Source	*Description*	*Storage Strategy*
Basic sensor data	Sensor name, deployment details, etc.	Relational- *MYSQL* database
Sensor measurement	Sensor measurements (text, images, etc.) .and time stamps	Non-relational *MongoDB*
Sensor properties	Accuracy, range, calibration details, etc.	Ontology file
Trustworthiness Calculations	The calculations, history of trust metric values	Relational- *MYSQL* database
Trustworthiness metrics	The complete list of calculated trust metrics	Ontology file

Table 3.1: Summary of the storage strategies used in WikiSensing

A typical user journey through the system can either begin at the client layer or the application layer. The client layer provides the WikiSensing functionalities via a web interface and the application layer via API Web services. The additional functionalities of geographical maps, Wiki pages, ontology query tools and messaging facilities are available when WikiSensing is accessed through the client layer. Both the web interface and the service API are connected to the application layer components through the *Application Logic Framework*.

When a user provides data to the system it is recorded in the database layer via the *Data Management* and then *Data Access* components. Conversely in the case of a query, data is fetched from the *Data Management* component using the *Data Access* component and returns the results to either the client layer or sends an *HTTP* response in the case where the request is invoked using API services.

3.2.4. The Data Model for Sensor data

The hybrid data storage model of WikiSensing is depicted by the *ER* (Entity Relationship) diagram in Figure 3.2. This model represents the storage for the sensor data and also includes the trustworthiness data and the wiki data (denoted using dashed lines). The trustworthiness data storage model (discussed in section 5.4.1) and the Wiki data (controlled by the Media wiki software) are included here to demonstrate their relationships with the sensor data. The WikiSensing data storage comprises of a relational *MySQL* database to store the sensor environment, virtual sensor configuration and user information, a relation free *MongoDB* stores the data points (sensor measurements) and time stamps of the data streams and an ontology to store the sensor property details (e.g. calibration, accuracy, range, etc.)

Relational Tables

The *Sensor Environment, Sensor Network, Data Stream, Trigger, Virtual Sensor Map, Virtual Sensor Query, Unit of Measure* and *User* are relational tables in *MySQL*. The *Sensor Environment* table stores the physical (or virtual in case of virtual sensors) representation of the sensor that contains basic information such as

sensor name, access rights, geography of location, deployment details, etc. Sensor environments are specialised into physical sensors and virtual sensors. Virtual sensors contain specific details on whether the storage of the aggregated sensor measurements are persistent or calculated dynamically. The information of the contributing sensors of a virtual sensor is maintained in the *Virtual Sensor Map*. This contains an identity of the virtual sensor and the list of identities of the contributing sensors. The sensor environment is linked on a one-to-one relationship with the sensor wiki information controlled by the *Media Wiki* software.

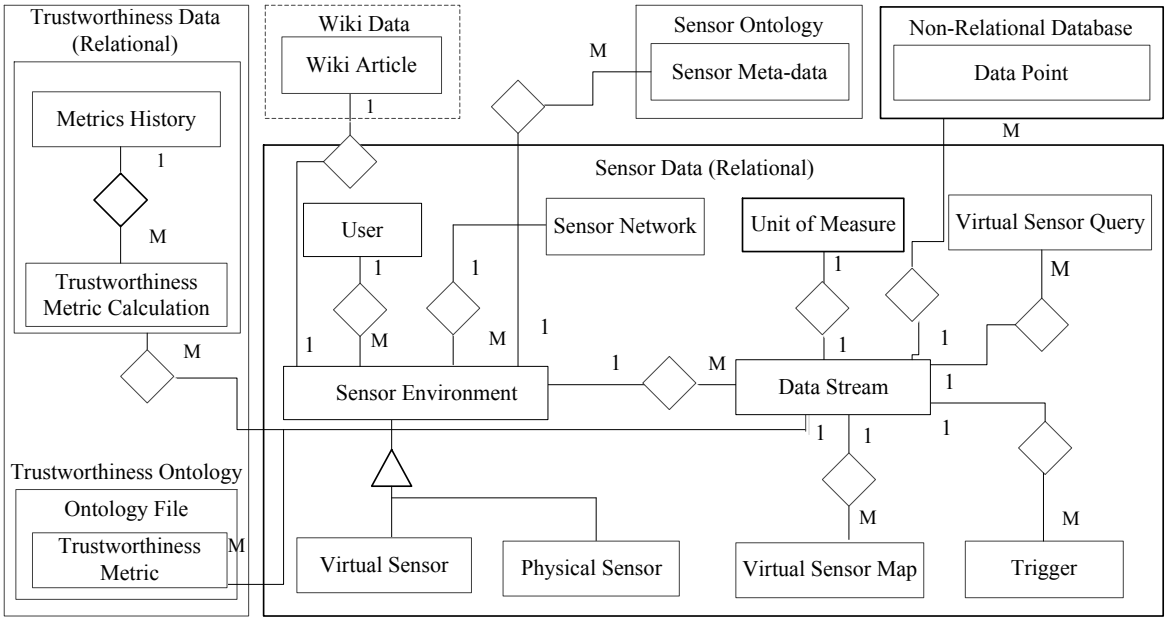


Figure 3.2: Entity Relationship Diagram of Data Model

A *Sensor Environment* can have multiple *Data Streams* for example, a pollution sensor measuring the CO_2, SO_2 and NO_2 values of the atmosphere. Moreover the *Data Stream* table maps to the actual sensor measurement stream that contains the type and reading information of that device. A single data stream can have multiple triggers imposed on it and this information is stored in the *Trigger* table. Trustworthiness data is calculated for sensor measurements and is linked with the *Data Stream*. The *Sensor Network* table contains the details to group a set of sensors that belong to a specific sensor network. The *User* table stores the details

of all active users of the system (e.g. login details, qualification, etc.). The table *Unit of Measure* contains the predefined measurement units as well as the units defined by the users. This also maintains conversion functions needed to calculate measurement units into a base unit of measurement.

Ontology

The sensor meta-data that consists of the sensor attributes (properties, characteristics, etc.) are maintained using an ontology. This enables the storage of extensible information as there are diverse types of sensors with different attributes that define various properties and capabilities of a sensor. The other advantages of using an ontology is that it enables the sharing of a common understanding as well as make domain assumptions explicit on the sensor attributes among people or software agents. For example, the same sensor properties may be named differently among people and the ontology is capable of expressing these as the same. An existing ontology (e.g. *OntoSensor* ontology) that contains schemas for sensor attributes is used as well as extended when necessary to incorporate additional attributes. The sensor information stored in the ontology is tagged as sensor attributes that are based on sensor properties, sensor specification details and user defined values. Users are free to add their own new attributes and this information is tagged in the ontology as a user defined value or property.

Non-Relational Data

The non-relational data is stored in a relational free *MongoDB*. *MongoDB* stores this information as a collection which is analogous to tables in a relational database. This is the *Data Point* information that contains the sensor measurements and corresponding time stamps. The *Data Point* also contains uniquely generated key and includes the environment identity and the data stream identity to establish a link with the entities in the relational database.

3.2.5. WikiSensing Query Constructs

The WikiSensing query language selects data from a combination of relational (*MySQL*), non-relational (*MongoDB*) and ontological data. The constructs that are introduced are prefixed with the term wiki for distinction. The query language is *SQL* like and is implemented in the *Data Management* module of the application layer. The following *SQL* code illustrates a sample structure of a query that is supported by WikiSensing. This example cites the constructs that are newly introduced with the '< >' tags used for denoting the input parameters.

```
SELECT [DISTINCT | ALL]
    column_expression1, column_expression2, ....
    [ FROM from_clause ]
    [ WHERE where_expression ]
    [ WIKI_LOCATION <Coordinates: Longitude, Latitude>]
    [ WIKI_RADIUS <Distance specified in kilometers or meters: km|m> ]
    [ WIKI_WINDOW <Window specified by time OR Number of readings h|m|r>]
    [ WIKI_UOM <Converts to standard unit of measure> ]
    [ WIKI_PROPORTION <Distance OR Time> ]
    [ WIKI_SAMPLE_STREAM ]
    [ WIKI_CONTINUE_FOR <Time specified in hours or minutes h|m> ]
```

The *FROM* clause selects the data stored in the relational, non-relational databases and ontology files. The functionality for selecting the data from these different sources resides in the application layer so that the users are not aware of such storage variation. The heterogeneous data sources are represented as relational tables to maintain a similar completion on querying from a single relational database. This is further exemplified in section 4.4. The construct *WIKI_WINDOW* indicates a window for the sensor readings specified either using a time unit (hours or minutes) or a record size. Specifying a time unit selects the readings within a specified time period prior to the execution time and specifying a record size selects the stated number of the latest sensor readings. *WIKI_PROPORTION* construct is used to indicate that the aggregated values must be based on the weighted mean of the specified attributes. The system currently supports linear aggregations such

as averaging and summing of data streams. The time frame or the distance or both can be specified with this construct to obtain a weighted mean. The *WIKI_LOCATION* construct select records within a location or more specifically within the specified coordinates. *WIKI_RADIUS* can be used in conjunction with the *WIKI_LOCATION* construct to specify a radius so that it selects records within a particular radius (in meters) to the specified location or coordinates. The *WIKI_SAMPLE_STREAM* construct samples the data streams to match the stream with the largest frequency when aggregating multiple data streams. *WIKI_UOM* is also used in aggregation queries to specify the base unit of measure. A query that contains the construct *WIKI_CONTINUE_FOR* returns values continuously for the specified time period and is used to obtain real-time data. Examples of the usage of these constructs are described in detail in section 4.4 page 75.

The backend or implementation of these constructs can either be developed from scratch or developed in conjunction using third party components. For example, stream processing engines such as *Esper* [29] can be used to facilitate queries that return data continuously or in the case where row or time windows are required. The motivation of developing the constructs from scratch is based on learning the dynamics needed to implement such features. However, as a next step it is more practical to use specialist external components (e.g. *Esper*) to handle the processing of certain functionalities.

3.3. Infrastructure for Trust Data Management

The infrastructure for the trustworthiness assessment requires capturing and storing of trust data. Furthermore the WikiSensing architecture (section 3.2) [8] is extended to support this generic framework by introducing new components highlighted in bold in Figure 5.4, Page 114. It must be noted that the components themselves are implemented in a generic way and can be accessibly plugged into a sensor data management system other than WikiSensing.

The architecture is based on a layered model with a data layer that includes a database for trustworthiness data. The application tier contains two sub layers, a business logic layer and a data management layer. The metric calculation is done in the *Assess Trustworthiness* component and is invoked by the *API* services. The trustworthiness management framework is discussed in detail in chapter 5 Modelling and Managing Trustworthiness.

3.4. Related Work

The centralised data storage design strategy of WikiSensing compares with the *Xively* [9] system. The advantage of centrally storing the data is that the system has more control over the data as it does not require communicating with sensor devices to obtain data for processing e.g. aggregation queries. However this contrasts with systems such as *Cougar* [20] that follows a distributed storage approach or *TinyDB* [17] that introduces special querying techniques to conserve the sensor battery life or adapt to limited network bandwidth. WikiSensing does not focus on these problems that are based on data acquisition but focuses mainly on the storage, querying and processing of sensor data from the point where data is acquired by the system.

3.5. Conclusion

The chapter demonstrated the design for a collaborative sensor data management system with trustworthiness assessment to address some of the challenges of Big data. The strategies for storage, querying and the organisation of data are key aspects addressed by this three tiered architecture design. The hybrid data storage model, novel query constructs and the usage of an ontology to represent data are some of the specific design approaches explained. The main functionalities of the system are designed into independent components to enable interoperability. The next chapter describes the implementation of some of the design features that were discussed.

4. Implementing WikiSensing's Data Management and Collaboration

The collaborative sensor data management system of WikiSensing is of central importance to the research work described in this thesis. It provides the services for sensor data management as well as support the trustworthiness management of that data (discussed in chapter 5). It is also the main source of sensor data and collaborative information that is used throughout this research. Moreover in the context of collaborative sensor data management, the accessibility and efficiency of the framework implementation is of high importance.

WikiSensing provides basic sensor data management features as well as specialised functionalities to create virtual sensors based on data aggregation. It contains a service layer so that these functionalities and data can be accessible from different programming technologies (e.g. C#, Java, Python, etc.). It also offers a *Wiki* website for users to annotate and share information on sensor data to enable online collaboration. Furthermore WikiSensing's hybrid storage model provides an efficient storage system to manage large volumes of sensor data.

The architecture of WikiSensing discussed in the previous chapter enumerated the main modules and strategies needed for collaborative sensor management. This chapter describes the implementation details of some of the main data management features of WikiSensing based on this architecture. This includes WikiSensing's hybrid data storage, the functionality of virtual sensors, the *API* web services, online collaboration, basic sensor data management functionalities and the management of heterogeneous and large binary data.

4.1. The Hybrid Data Storage

WikiSensing's database is implemented using a hybrid storage strategy with a relational database, non-relational database and ontology to store different parts of the sensor data. Each of these databases run on separate virtual machines on the *IC Cloud* [85]. Furthermore multiple server instances are used as replicas for the non-relational database cluster that stores the sensor data streams. The hybrid database strategy stores the sensor measurements in a *MongoDB* non-relational database, the sensor meta-information (accuracy, range, etc.) using an ontology and all other sensor data (location details, virtual sensor, unit of measurements, etc.) in a relational *MySQL* database.

4.1.1. Relational and Non-Relational Databases

Data in *MongoDB* is stored in collections which is a grouping of *MongoDB* documents or records. A collection maps to the concept of an *RDBMS* table and documents within a collection can have different fields (records with heterogeneous formats). The sensor measurements in WikiSensing are stored in a single collection.

The *MongoDB* in WikiSensing is configured as a cluster of *shards* [83] as illustrated by Figure 4.1. Moreover *Sharding* is applied to the collection that stores the sensor measurements. *Sharding* leads to better performance and improved scalability to adapt with the increase in demand of users and storage space. Sharding is also known as horizontal partitioning. Hence when Sharding is implemented on *MongoDB* a replica of the schema is created, and then the data divided among each shard. This contrasts to vertical partitioning that splits up the data stored in one entity into multiple entities.

The *Sharding* process enables data records to be stored across multiple machines and is the mechanism used by *MongoDB* to support the growth of data. With the increase of the size of sensor data (e.g. with the growth of new sensors and measurements), multiple machines are required to store this data and provide

acceptable read and write throughput. With *Sharding* WikiSensing is able to scale horizontally by adding more hardware to support data growth and the demands of read and write operations. Moreover when the data capacity reaches a certain threshold *Sharding* can be applied to add new storage resources. This threshold is based on the usage of disk space in a machine and is currently, heuristically set to be between 60% and 70%. The *MongoDB* cluster automatically corrects imbalances between shards by migrating ranges of data from one shard to another.

Indexes are used in *MongoDB* for efficient execution of queries that are fundamentally similar to other database systems. All *MongoDB* collections have an index on the *_id* field (an id representing the document id, automatically generated from *MongoDB* for each record) that exists by default. There is also a user defined index created based on *UserId* and *SensorId*. Usually any sensor measurement that is stored in WikiSensing contains these fields and is also used when querying.

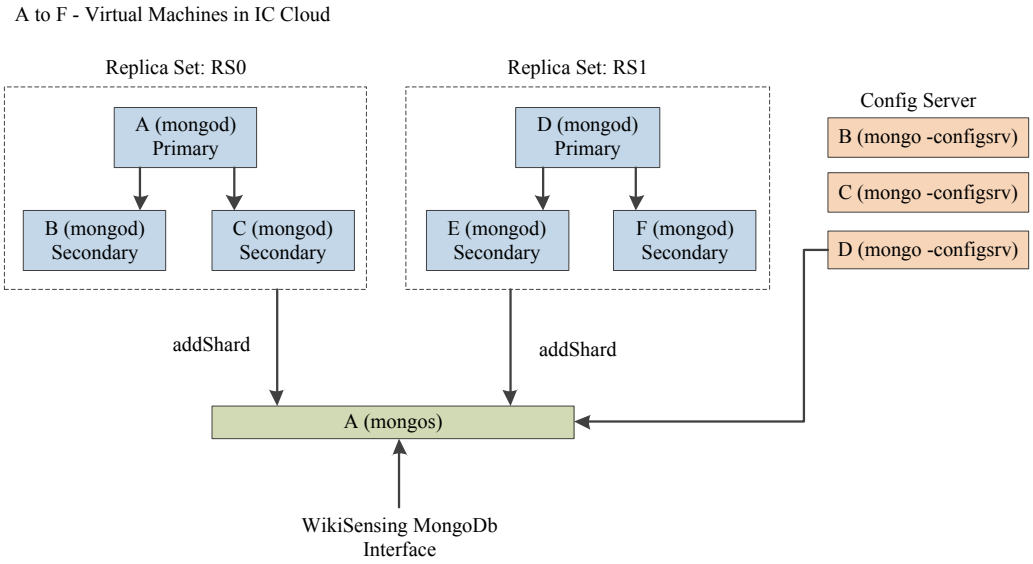


Figure 4.1: The *MongoDB* cluster for WikiSensing using *Sharding*

The relational database of WikiSensing is a *MySQL* server deployed on a virtual machine in the *IC Cloud*. WikiSensing's application layer communicates with all database servers through a secure virtual private network. Moreover the connection and communication between the data from other sources such as the

non-relational database and the ontology is mapped by the business logic at the application layer.

4.1.2. Managing Data by Ontology

The use of an ontology to store the sensor properties or meta-information enables a shared understanding of the concepts in the domain of sensors. Sensor properties can be extensible as well as different users can have several interpretations of certain concepts. Hence an ontology resolves these issues by supporting an extensible list as well as providing constructs (*RDF and OWL*) to incorporate semantics into the data. The advantage of using *RDF* (Resource description Framework) [86] with *OWL* (Web Ontology Language) [67] over *XML* is based on the complexity of querying the data. There are a large number of ways in which data can be represented in *XML* and hence it is difficult to design queries that are independent of this structure. In contrast *RDF* enforces a standard way of writing statements so that irrespective of the way it occurs in a document, they produce the same effect in *RDF* terms (e.g. *RDF* uses *URI's* to represent elements that are globally unique). WikiSensing uses the *OntoSensor* ontology to represent sensor properties as well as extending this ontology in situations where new properties are introduced.

The ontology data is managed by using the proprietary *Virtuoso* server. Moreover the *dotNetRDF* [87] *.Net* Library is used to access (e.g. query, update, etc.) the ontology data from the WikiSensing middleware. *dotNetRDF* provides a set of *API's* for working with ontology files and supports the *Virtuoso* data store.

4.1.3. Motivations for Hybrid Data Storage

The hybrid database approach of WikiSensing offers several advantages. Firstly using a high speed database to store the vast number of sensor readings will enhance performance of the data access. *MongoDB* is a document-oriented, schema free storage system that uses memory-mapped files [23]. It is also a relational free

database that provides better performance to a relational database such as *MySQL* [88][24]. The schema-less nature of *MongoDB* has the advantage of storing heterogeneous types of records. Moreover sensors that produce different types or different numbers of measurement can be accommodated in a single document (analogues to a relational table).

Secondly non-relational databases such as *MongoDB* lack the atomicity, consistency and durability transaction properties [25] and are not suitable to store information that require a degree of concurrency control. Hence the primary aim of using *MongoDB* is that it is lightweight and fast as it does not use traditional locking or complex transactions with rollback [26]. Furthermore *MongoDB* is used to store data that is not modified but only inserted and hence does not require any data record locking mechanisms. On the other hand data that are usually modified (e.g. sensor environment details, virtual sensor configurations, etc.) are stored in the more mature *MySQL* database that ensures these transactional properties.

Thirdly the use of the ontology enables the storage of extensible information while preserving a common vocabulary. For example, ontologies help to distinguish different sensor properties (e.g. sensor resolution and sensitivity) as well as symbolise properties that are the same but named differently (e.g. sensor accuracy or true variation). It is also useful to define rules by setting relationships between these concepts, for example subclasses are a good way to define certain sensor properties that have similar characteristics.

4.2. Virtual Sensors

The rationale and the practical usage are described to understand the motivation and requirement for virtual sensors.

4.2.1. The Rationale

When the sensor data are not sufficient, or when a direct sensor measurement at a specific location is missing, virtual sensing is adopted. A virtual sensor is a sensor that is not physically deployed at a certain location but uses data streams of nearby located sensors to provide sensor measurements. Virtual sensors are implemented by selecting a set of contributing sensor data streams, either by using the web interface or the application services. The selected streams are then aggregated to provide measurements of the virtual sensor. In such cases these aggregations are operations that produce a single value of a data stream. Hence virtual sensors are an extremely useful feature that provides sensor readings in the case where no physical sensors are present at specific locations. Moreover it is also useful in situations where a low quality sensor may be physically deployed; however aggregating a set of high quality sensors that are deployed nearby can result in better measurements.

4.2.2. Practical Usage

In practice, existing sensor data is used to create this conceptual item of a "virtual sensor". Usually this requires the knowledge and experience of the collaborating users for example, the knowledge on geographical locations, the reliability factors of the sensor devices, etc. Hence the knowledge of the collaborators is useful to annotate sensors so that they can be selected rationally in order to create virtual sensors that can produce acceptable measurements.

Figure 4.2 illustrates a scenario where several physical sensor devices are combined to create a virtual sensor. During the first stage of this process the collaborating users are involved in annotating the sensor streams with geographical information and sensor meta-data such as reliability, precision and accuracy. This information is recorded in the wiki. The second stage involves the users selecting the physical sensors that would contribute to the virtual sensor. The calculated measurements of virtual sensors can either be stored in the database or produced

dynamically when needed. The query in section 4.4.3 of page 83 (stage 4) used in aggregating data streams for the virtual sensor can be updated to increase or reduce the scope of the sensor (e.g. change radius) or modified to change measurement window size.

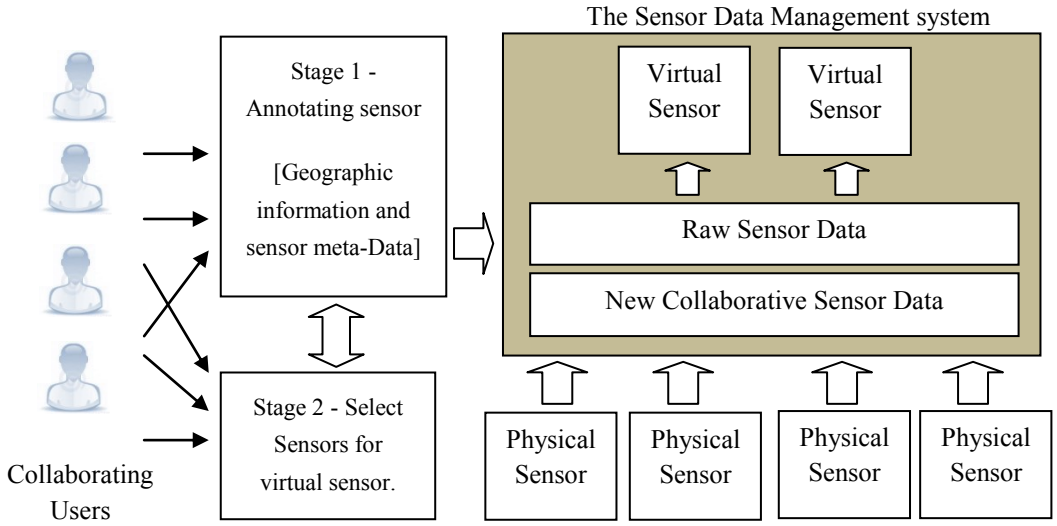


Figure 4.2: Collaborating sensors to create virtual sensors

The *Data Aggregation* module of the WikiSensing architecture contains the functionality to implement virtual sensors. These functionalities include registering virtual sensors, querying and selecting contributing sensors as well as generating the aggregate queries. The *API Web Services* component is used to connect the contributing sensors to acquire the sensor measurements needed for the virtual sensor. These readings are then aggregated and if requested stored in the database using the *Data Access* component.

4.3. Collaboration

The basis of how collaboration is enabled and how this data maps to sensor data is explained to understand the implementation of collaboration in WikiSensing.

4.3.1. The Rationale

The success of a collaborative system is centred on the usability and the organization of the information [89]. In order for users to collaborate, the system must have an infrastructure in place to support and facilitate the sharing of knowledge and information. Hence WikiSensing uses the popular *MediaWiki* (*www.mediawiki.org)* framework to support with online collaboration. The Wiki pages for online collaboration runs on the client layer that are hosted using the *MediaWiki* deployed in the application layer of the WikiSensing architecture. The goal of collaboration is to obtain annotations, comments as well as ratings on the sensor information by users with different areas of expertise.

4.3.2. Collaborative Data

There is a clear distinction of the information in WikiSensing between a collaborative data layer and a data management layer. The collaborative data layer sits on top of the data management layer (Figure 4.3) with all sensor information in the underlining data layer having a mapping on to the Wiki layer. This is implemented by automatically creating and updating Wiki pages when new sensors get registered as well as when the information is updated. This enables the transparency of the data so that the users can annotate and comment on up-to-date information such as the sensor environments, sensor meta-data, data streams, virtual sensors, etc. that are managed in the underlining sensor data management layer.

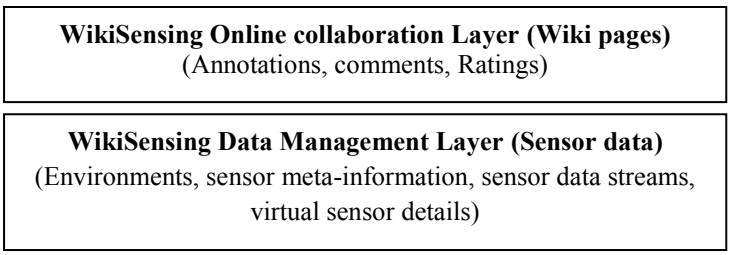


Figure 4.3: The WikiSensing Information Layers

Further details on how Wiki pages are created, mapped with sensor data and how users annotate information in WikiSensing are discussed in the following section.

4.4. Basic Sensor Data Management Components

This section presents six basic scenarios of sensor data management functionalities in WikiSensing. The first three scenarios describe the organisation of information, the aggregation of multiple sensor data streams and the creation of virtual sensors. These functionalities are based on sensors producing sequential data e.g. sensor measurements and time stamps. The fourth scenario describes heterogeneous data management, the fifth explains how WikiSensing handles large binary data objects such as images, and the sixth describes WikiSensing API web service components.

4.4.1. Organising Sensor Information

Stage 1: Registering an Environment for a sensor

The first mandatory step for registering sensors is to create an environment that the sensor is deployed in. This information (Table 4.1) includes location details, (e.g. name of location, city, street, country, etc.) as well as geographical coordinates (e.g. longitude, latitude, etc.) that can be selected using *Google Maps*. Moreover information on the nature of the sensors (disposition, exposure, etc.), the sensor network name (if sensor is member of a sensor network) are also recorded.

The users are encouraged to provide a feed or data stream description that contains the type of sensor (e.g. temperature sensor, accelerometers, pollution sensors, *GUSTO* sensor [22], etc.). The accessibility of the sensor data can be set as private so that it is only visible to the creator or set as public which makes it accessible to any user of WikiSensing.

Field	*Mandatory*	*Domain*	*Description*
Sensor identity	Yes	Number	The identity of the sensor
Environment name	Yes	String	The name of the environment that the sensor is deployed
Feed description	No	String	Description of the data streams
Location description	Yes	String	Description of the deployed location of the sensor
Access right	No	Boolean	Public or private, and private by default
Latitude	No	Float	Latitude of the sensor environment
Longitude	No	Float	Longitude of the sensor environment
Elevation	No	Float	Elevation of the sensor
Exposure	No	String	Whether the sensor is deployed indoor or outdoor
Disposition	No	String	Whether the sensor location is fixed or mobile
Domain	No	String	Whether the sensor is physical or virtual
Virtual sensor data persistence	No	Boolean	Whether the virtual sensor readings are stored or generated dynamically
Sensor network	No	String	The network Identity of the sensor
Data stream identity	Yes	String	The identity of the data stream
Stream type	Yes	String	The type of attribute that is measured
Unit of measure	Yes	String	The measuring unit of the data stream

Table 4.1: The list of fields involved in registering sensors in WikiSensing

Stage 2: Registering the data streams of a sensor

Sensor devices can measure several attributes of an environment and produce multiple data streams. For example, a *GUSTO* sensor can measure the *NO, NO_2, SO_2* and *ozone* air pollutant readings and provide four different data streams. Hence data streams are representations of a physical or virtual sensor that is deployed at some location. The *data stream* usually contains a *sensor type* and a *unit of measure*.

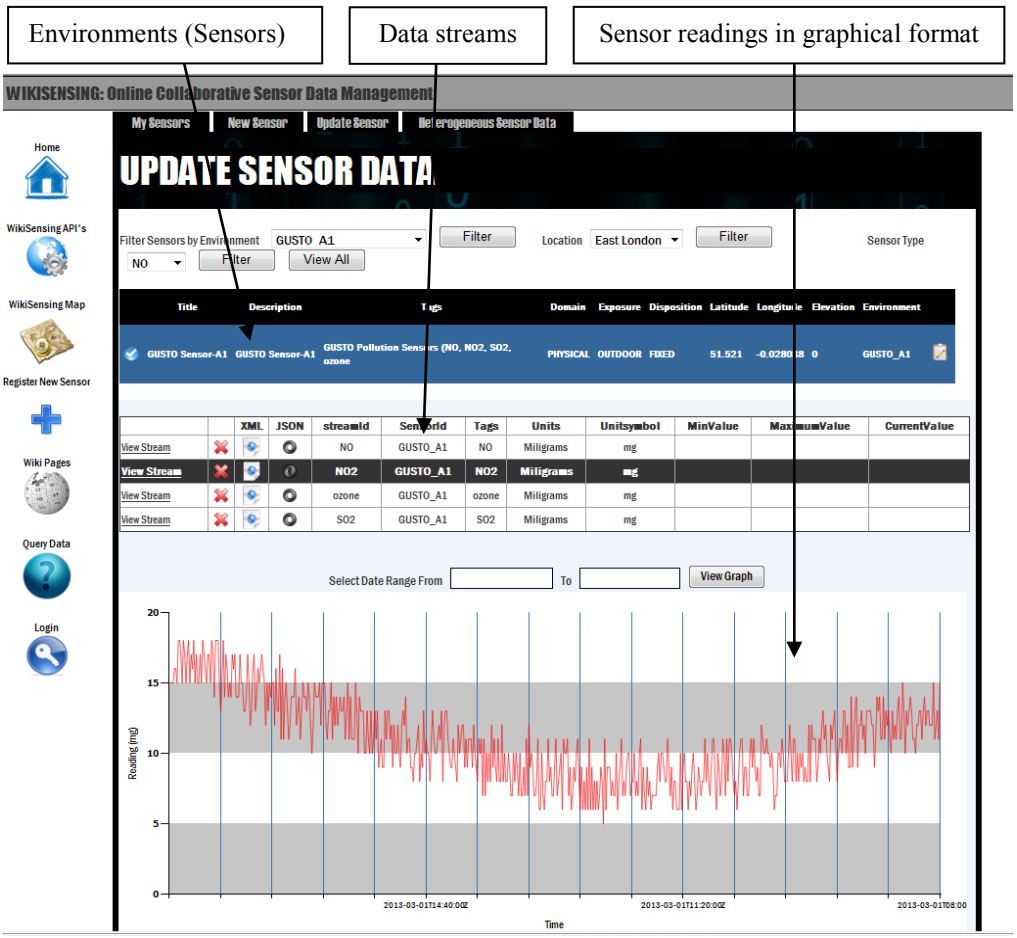


Figure 4.4: WikiSensing graphical view of sensor data streams

The measurement units for a data stream can either be selected from a predefined list or can be explicitly specified by a user. When defining a new unit of measurement users are required to provide a conversion function to a base unit. Once an environment (deployed sensor) has been defined and data streams attached to it, data points or measurements can be added. The *data point* consists of sensor *measurements* and *time stamps*. The users can also automatically connect the sensor data streams to the system via the web service layer. This is done by obtaining a service reference of the WikiSensing web services and can be done using any programming platform as explained in section 4.4.6. The data stream information can be viewed graphically as illustrated in Figure 4.4.

A *Wiki* page representation (Figure 4.5) of the sensor is created automatically when sensors are registered. This page contains a description of the sensor environment followed by its details of the data streams.

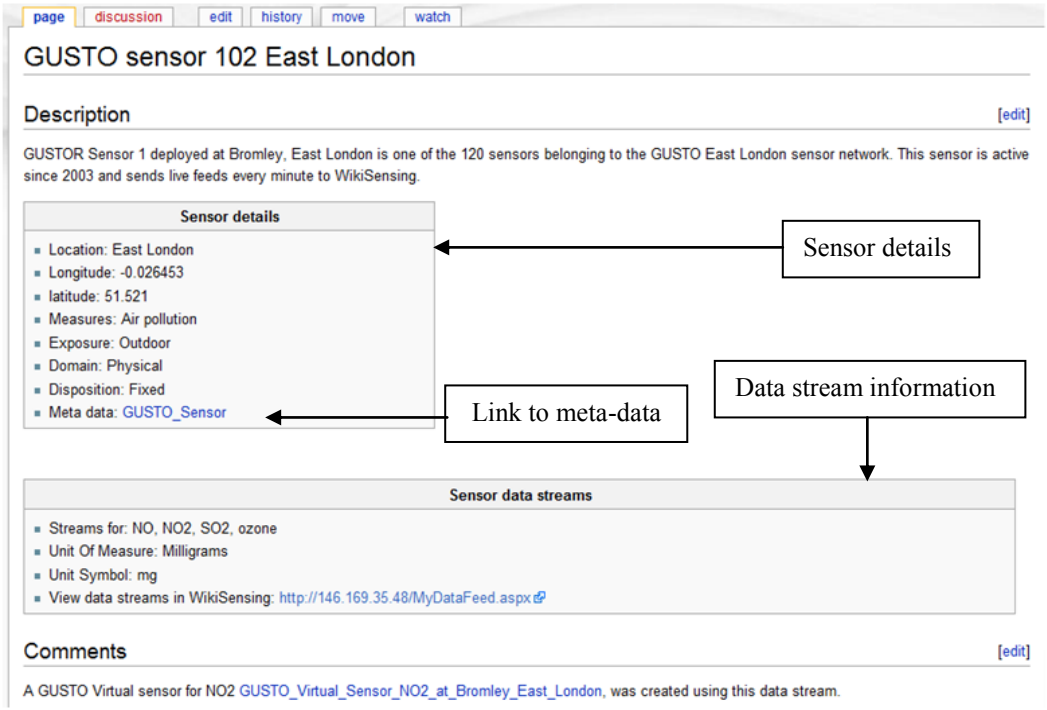


Figure 4.5: Wiki pages that record the sensor and data stream information

The system also automatically links the environment with a page that contains the relevant sensor meta-information (Figure A.1, of Appendix). The *Wiki* page containing sensor meta-data lists the sensor properties and features that can also be updated by collaborating users. If needed users are able to create new sensor meta-data *Wiki* pages in case where a matching page does not exist. These *Wiki* pages are automatically updated when corresponding information on the system are modified by the user.

At the bottom of the *Wiki* page displayed in Figure A.1 in the Appendix, shows a reference to substantiate the information added to the page by the user. In this example the user annotates a *GUSTO* (*Generic Ultraviolet Sensor Technologies and Observations*) sensor by referencing research work [22]. This is

considered good practice to show provenance for the annotations added by users as experienced with other wiki websites such as *Wikipedia*.

Stage 3: Query sensor the data streams

The following sample query averages readings of a single sensor for a window size of 1 hour. The *WIKI_WINDOW* query construct indicates a time window to select sensor measurements within an hour prior to the execution time. This can also be specified using the number of measurements, which selects the preceding records from the current time stamp.

```
SELECT Average (p.measurement)
    FROM Environment e, DataStream d, DataPoint p
    WHERE e.sensorId = 'GUSTO_A1'
    AND d.sensorType = 'NO2'
    WIKI_WINDOW    = 1<h>
```

Environment and *DataStream* are relational data tables and *DataPoint* represents the data from the non-relational database. However the *DataPoint* is represented as a relational table for the convenience of validating the query (excluding *WIKI* prefixed constructs) as well as to preserve the *SQL* like query structure. Moreover explicit *SQL* joins are not required to obtain the correct data as the joining is implemented in the application middleware.

Stage 4: Registering a sensor network

A sensor network is a group of (usually homogeneous) sensors deployed at multiple locations providing data streams that can be aggregated to obtain a set of combined sensor readings.

Creating a sensor network in WikiSensing involves two main steps. The first step is to register the sensor network by providing the details that are listed in Table 4.2. The second step is to reference the sensor network from member sensor environments using the *Sensor Network Id*. A *Wiki* page is automatically created for the sensor network listing its member sensors.

Field	*Mandatory*	*Domain*	*Description*
Sensor Network Id	Yes	Number	The identity of the sensor network
Sensor Network Name	Yes	Number	The name of the sensor network
Description	Yes	String	A description about the sensor network
Purpose	No	String	The motivation for creating a sensor network

Table 4.2: The list of fields to register a sensor network

Stage 5: Registering sensors to a sensor network

Firstly the user has to create the set of sensors individually by repeating the steps (1 to 4) of the functionality in section 4.4.1 specifying the *Sensor Network Id*. This links the sensors with the sensor network. The relevant sensor network Wiki page is then updated with this information.

Stage 6: Query sensor data in a sensor network

The following sample query aggregates a set of sensors that belong to a particular sensor network.

```
SELECT Average (p.measurement)
   FROM Environment e, Datastream d, DataPoint p
   WHERE e.sensorNetwork = 'GUSTO Sensor Network-1'
   AND d.sensorType = 'NO2'
   WIKI_WINDOW = 1<h>
```

Stage 7: Policies for managing historical sensor measurements

There are two policies used in WikiSensing to manage historical data. The first policy maintains historical data in storage until a user specified time period (e.g. 30 days) with a maximum time period of 90 days. The second policy or default policy aggregates (e.g. averages) sensor data after specific time period (e.g. 7 days) and records a single value. Moreover this time period can be specified by the user with a maximum time period of 90 day being set by the system. However the limitation of the second option is that it requires sensor measurement to be in numeric format.

4.4.2. The Aggregation of Multiple Data Streams

Stage 1: View sensor data streams

When users log in to WikiSensing they are able to view a list of sensors and sensor networks that were created by them as well as all sensors and sensor networks that are registered as public. Furthermore users are able to view the data stream of these sensors as well as request for aggregated measurements.

When for example, the average temperature of *South Kensington, London* is requested by specifying coordinates (e.g. longitude and latitude) the relevant sensor data streams are aggregated to produce measurements. Moreover the system checks if potential sensor data streams are compatible for aggregation (e.g. same type). If compatible they are then checked for other disparities as data streams produced by different sensor devices may have different characteristics, for instance different output frequencies or different units of measurements.

Stage 2: Convert to a single unit of measurement

When the units of measurements are different, WikiSensing automatically converts the values of the data streams to the unit of measure that is used by the majority of the data streams. If there are the same numbers of data streams with different units the system would then use a default unit of measurement. These rules are overridden when the user explicitly specifies a unit of measurement in the query using the *WIKI_UOM* construct.

Stage 3: Sample different frequencies of data streams

There are two policies to handle disparity of frequency among data streams. The first policy samples the time frames of the data stream to fit the stream with the largest time interval. Table 4.3 illustrates this by combining the first stream's readings at 10:27:30 and 10:28:0 to a single time frame of 10:28:0 so that it can be accurately mapped with the frequencies of the second data stream. This policy is applied when the user explicitly specifies the *WIKI_SAMPLE_STREAM* construct in the query.

Frequency of submitting readings every 30 seconds	10:27:30	10:28:0	10:28:30	10:29:0	10:29:30	10:30:0	10:30:30	10:31:0
Frequency of submitting readings every 60 seconds		10:28:0		10:29:0		10:30:0		10:31:0
Sampled frequency of aggregated stream		10:28:0		10:29:0		10:30:0		10:31:0

Table 4.3: The sampling of the frequency of multiple data streams

The second, or default, policy is applied when the user does not specify any construct in the query. It individually averages the data streams of each sensor disregarding the differences of the frequencies. For example, it selects the measurement within the specified time range and combines (e.g. average) these values.

Stage 4: Aggregate Queries

The following query outputs the average temperature reading at location with coordinates 51.521 and -0.026453. The *WKI_PROPORTION_ON* construct is used to indicate that the aggregated measurements are based on the weighted mean of the specified attributes (in this case the distance from the specified coordinates). The *WIKI_LOCATION* construct selects records within a location specified or the geographical coordinates. This query can be further extended using the *WIKI_RADIUS* construct that selects sensors within a radius (specified in kilometres) to the specified location or coordinates. The sensors within the specified radius are selected using the *Haversine formula* [90]. This formula provides the great-circle distances between two points on a sphere using the longitudes and latitudes.

```
SELECT Average (p.measurement)
    FROM Datastream d, DataPoint p
    WHERE d.sensorType = 'NO2'
```

```
WIKI_LOCATION    = <51.521, -0.026453>
WIKI_RADIUS   = 0.25<km>
WIKI_WINDOW   = 1<h>
WIKI_UOM      = <Celsius>
WIKI_PROPORTION = <DISTANCE>
WIKI_SAMPLE_STREAM
```

The *WIKI_SAMPLE_STREAM* construct samples the data streams to match the stream with the largest frequency (Table 4.3). The user has the option to specify this query as continuous query with the construct *WIKI_CONTINUE_FOR <time interval in hours or minutes>*. This enforces the query to produce outputs continuously for the specified time period.

4.4.3. Creating a Virtual Sensor

Virtual sensors are usually created when there is no physical sensor deployed at a specific location. This is also useful when users require the aggregation of several data streams to be persistent.

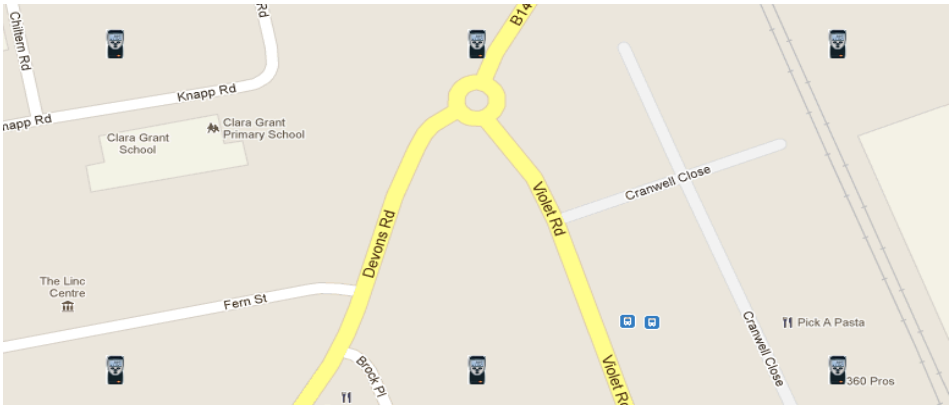


Figure 4.6: The WikiSensing map illustrating the deployment of sensors

Stage 1: The search phase

The users can either view the WikiSensing map or query to check the locations of available sensors. Figure 4.6 illustrates an instance of a map used in WikiSensing followed by an example query that would select available sensors in a specific location.

```
SELECT e.sensorId
    FROM Environment e, Datastream d
    WHERE d.sensorType = 'NO2'
    WIKI_LOCATION    = <51.521, -0.026453>
    WIKI_RADIUS      = 0.25<km>
```

This query selects sensors that measure the air pollutant NO_2 within a radius of 0.25 km of the location specified with the coordinates 51.521 and -0.026453.

Stage 2: Registering a virtual sensor

If the user requires sensor measurements from a particular location where a sensor is not physically deployed the user can create a virtual sensor. This is done by specifying its details similar to registering a regular sensor described in scenario 1 with the exception that the *domain* field is set as '*virtual'*. In addition users can specify the v*irtual sensor data persistence* field (Table 4.1) to be either persistent or dynamic.

The two categories of virtual sensors are the ones which store the aggregated measurements (persistence) and the virtual sensors that generate measurements dynamically. The measurements of persistent virtual sensors can be traced for the origins of the contributing sensor data streams. For example, in case where there are doubts on a virtual sensor, the data can be audited as its measurements are recorded. The audit can check for problems by analysing the history of streams that are included as well as removed from a virtual sensor. In contrast dynamic virtual sensors produce their reading on request, and their output is generated by aggregating the data streams in real time.

Stage 3: Select and record contributing sensors

The user can select a set of contributing sensors (usually sensors that are nearby) for the virtual sensor (Figure 4.7). In this example, sensor S_1 (at distance X) and S_2 (at distance Y) are selected for the virtual sensor VS. The user also has the

flexibility to add more sensors or remove existing contributing sensors from the virtual sensor.

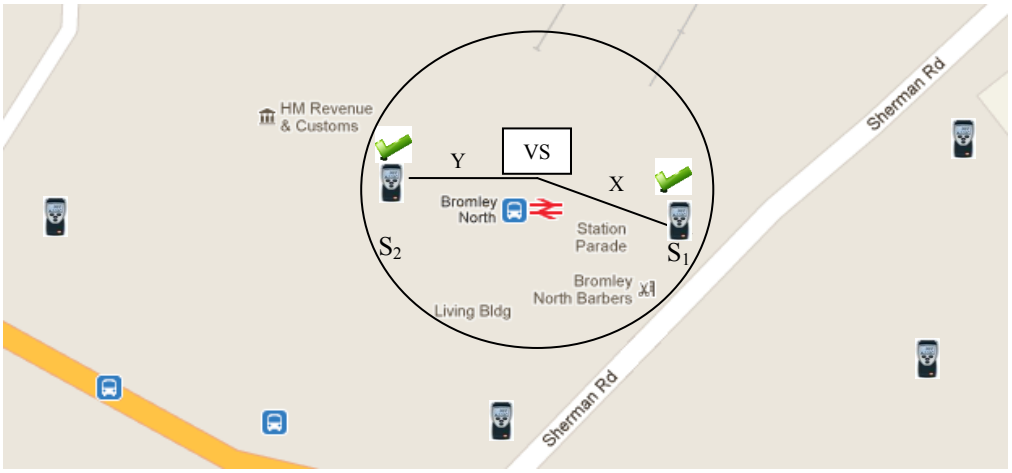


Figure 4.7: Selecting sensors to create a virtual sensor

The sensors that contribute to a virtual sensor are recorded in a virtual sensor map table, whose fields are listed in Table 4.4. The optimize column is updated when the user explicitly requests the selected contributing sensors list to be optimized. The system updates this column with virtual sensor identities (virtual sensors that are persistent) that are already created using a subset of the selected sensors. The aim is to reduce the database reads using existing virtual sensor data streams that are already formulated. Figure 4.8 illustrates the WikiSensing interface that enables users to add sensor data streams to a virtual sensor.

Field	*Mandatory*	*Domain*	*Description*
Virtual Sensor Environment Identity	Yes	Number	The identity of virtual sensor environment
Contributing Sensor Environment Identity	Yes	Number	The identity of contributing sensor environment
Data stream identity Virtual Sensor	Yes	Number	The identity of the data stream of virtual sensor
Data stream identity Contributing Sensor	Yes	Number	The identity of the data stream of contributing sensor
Optimize	No	Number	List of identities of selected virtual sensors that are used to optimize performance.

Table 4.4: The list of fields to register a virtual sensor network

It is assumed that the contributing sensor streams are of same type (e.g. measuring the same physical phenomena). Moreover it is also assumed that these sensors are continuously functioning and submit data consistently in accordance to its frequency. However it also expected that the user explicitly removes a contributing sensor from a virtual sensor when it no longer provides measurements or seize to function. A further discussion is done on assessing the trustworthiness of these contributing sensors of virtual sensors in section 9.2.2, page 196.

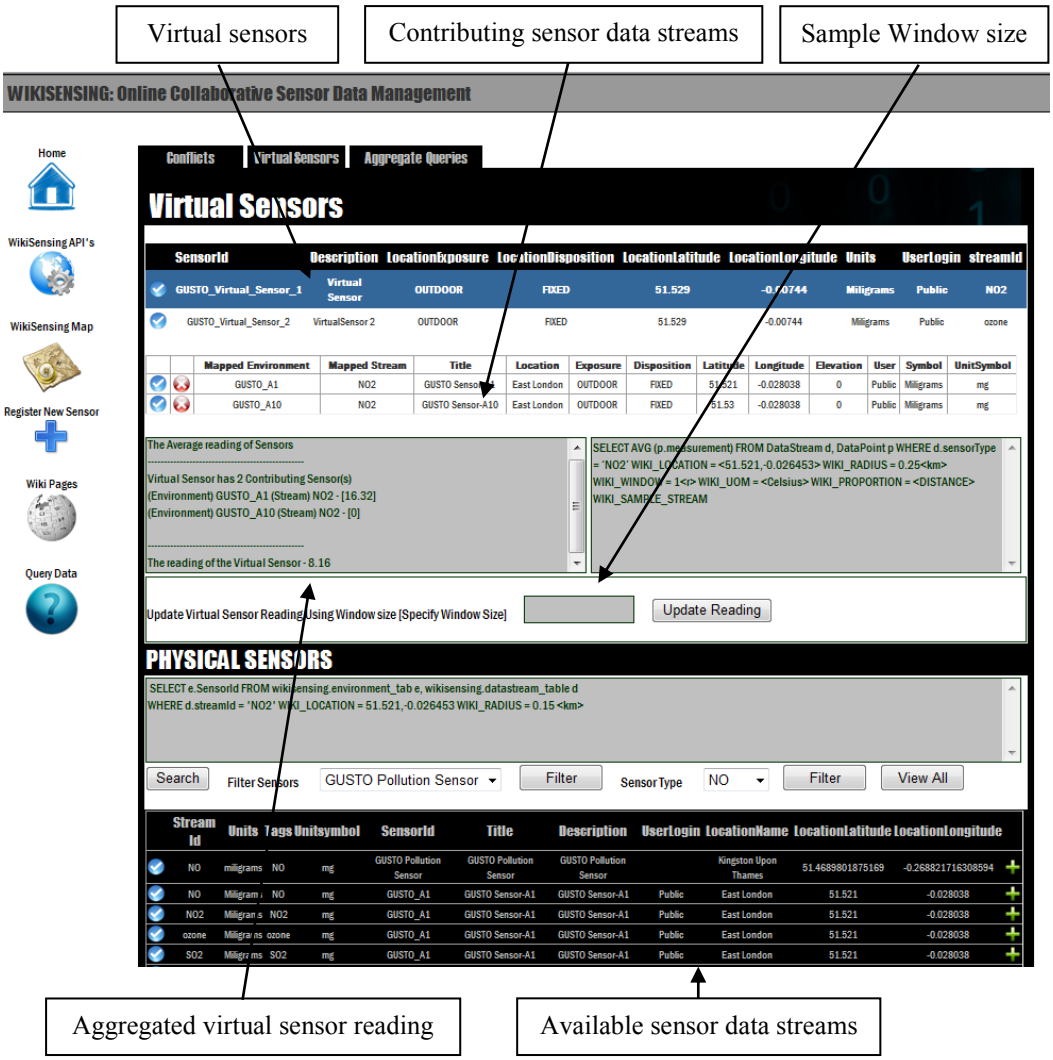


Figure 4.8: WikiSensing Interface for selecting sensor streams to create a virtual sensor

Stage 4: Aggregating the data streams of the contributing sensors

The system provides an aggregated sensor measurement (of selected sensors) as the reading for the virtual sensor. The following query is an example that aggregates readings for a virtual sensor.

```
SELECT AVG (p.measurement)
    FROM Environment e, DataStream d, DataPoint p
    WHERE d.sensorType = 'NO2'
    AND e.sensorId IN (<List of sensors selected by the user>)
    WIKI_WINDOW = 1<r>
    WIKI_UOM    = <Milligrams>
    WIKI_PROPORTION = <DISTANCE>
    WIKI_SAMPLE_STREAM
```

Figure 4.8 illustrates an aggregated measurement of the virtual sensor (*GUSTO_Virtual_Sensor_1*) of type NO_2 that consists of the contributing *GUSTO_A1* and *GUSTO_A10*. The construct *WIKI_PROPORTION* is an indication to aggregate the sensor streams based on a weighted calculation. This can be the weighted mean of the distance (formula 4.1) from the specified location or any other specified calculation.

$$\overline{X} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} \qquad (4.1)$$

For example, if a weighted calculation is used $\overline{X}$ would denote the weighted arithmetic mean with x and w being the values and weights of the items. The weight is the proportion to the spatial distances from the contributing sensor to the location of the virtual sensor. The aggregation query that is responsible for obtaining virtual sensor readings is stored in the virtual sensor query table (Table 4.5) with the ability to be modified on request.

Field	*Mandatory*	*Domain*	*Description*
Virtual Sensor Environment Identity	Yes	Number	The identity of the virtual sensor
Data stream identity Virtual Sensor	Yes	Number	The identity of the data stream of virtual sensor
Query	Yes	String	The SQL of the aggregate query

Table 4.5: The list of fields in the virtual sensor query table

When a user completes registration a Wiki page for the virtual sensor is automatically created and the provided information is recorded (Figure 4.9). The Wiki page also gets automatically updated when a user updates the composition of the virtual sensor.

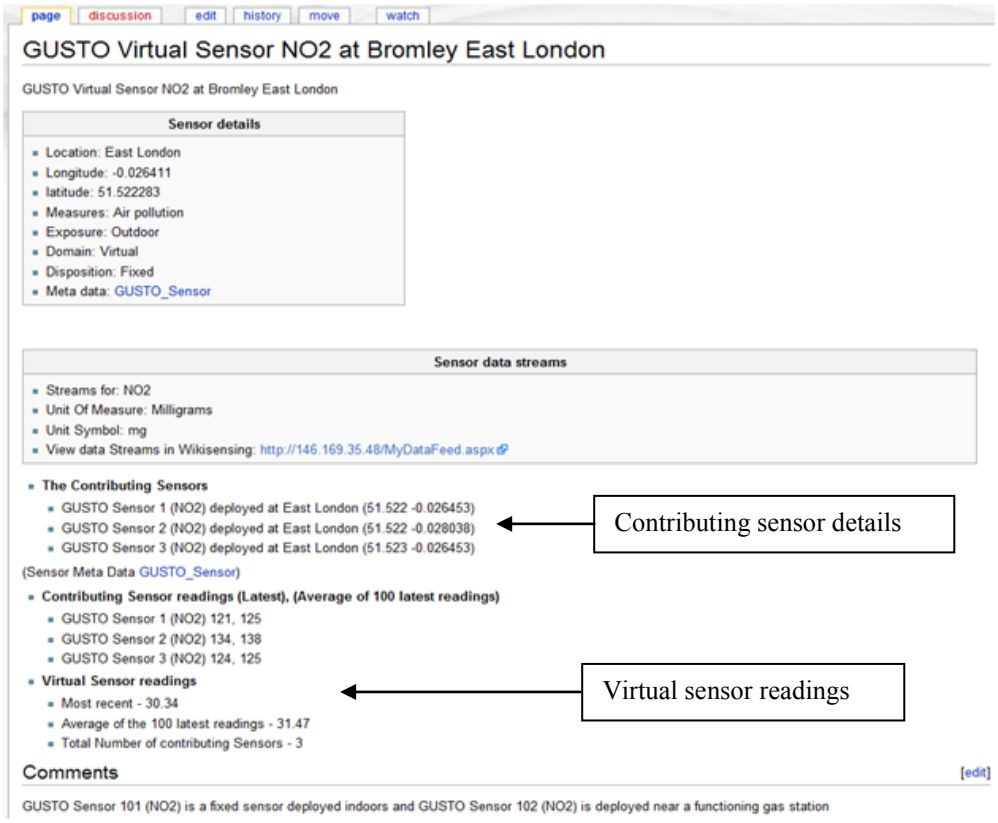


Figure 4.9: Wiki page recording information on a virtual sensor

4.4.4. Storing and Querying Heterogeneous Data

WikiSensing supports the storage and querying of heterogeneous sensor data records with variable formats. These features are supported through the web service *API*. Moreover heterogeneous data can also be queried using any combination of data fields of the sensor stream. This contrasts with the scenarios discussed previously, that dealt with homogeneous records with fixed formats.

An instance of the *sensorObject* class is used to specify the data structure that needs to be stored in WikiSensing. The *sensorObject* class is embedded as an extensible list so that the records can have any number of fields. The *sensorObject* comprises of a *field name* and *value* with the former representing the name of the field and the latter being the value of that field. For example the field name can be the '*Manufacture'* of the sensor and value can be '*Air Quality Ltd'*. Data to WikiSensing can be submitted using multiple *sensorObjects* within a *sensorObjectList.* Moreover this can be the more efficient option when compared to sending a single *sensorObject* per *HTTP* request. Hence data can be kept by the user instead of directly loading into WikiSensing. However it is the responsibility of the user application that submits data to WikiSensing to preserve ordering of measurement timestamps in order to maintain correct sequence of records.

The *sensorObject* is also used to construct query results. Similar to storing information, sensor data is dynamically mapped into this object and returned to the user. The query fields are again extensible with the user only needing to specify them in the *URL* query string [91]. The fields specified in the query is extracted and dynamically mapped to database fields. The following *XML* output illustrates a sample query result with the information encapsulated in a *sensorObjectList*. The *sensorObjectList* contains an extensible list of *sensorObject* instances.

```
<sensorObjectList>
   <sensorObject>
     <fieldName>TimeStamp</fieldName>
     <value>2013-03-19T10:15:30Z</value>
   </sensorObject>
   <sensorObject>
     <fieldName>Accelerometer Axis X</fieldName>
     <value>+1.25</value>
   </sensorObject>
   <sensorObject>
     <fieldName>Accelerometer Axis Y</fieldName>
     <value>+0.33</value>
    </sensorObject>
</sensorObjectList>
```

A limitation of a maximum 1000 output records is set by the system for a single query response. This limitation is imposed to control the data exchange between the WikiSensing web server and to prevent unnecessary overloading. This limitation can be surpassed programmatically from the client-end, when more records are required. The output data records are by default sorted by data and time. Hence a script on the client-end can obtain the oldest timestamp of the output record set and make more requests to get the subsequent set of records (e.g. a sliding window implementation).

4.4.5. Managing Large Binary Data

Image data are transferred to WikiSensing in the *Base64* [92] format. This format represents binary data as *ASCII* strings by translating it into a radix-64 representation. This is a popular format that is commonly used for encoding large amounts of binary data (e.g. images, video clips, etc.) that needs to be stored and transferred over the Web. WikiSensing converts the *Base64* representation of the image into an image format and saves it in the *MongoDB* as a *GridFS* [36, 37] object. The image is again encoded back into *Base64* format and transferred in *XML* or *JSON* when the information is queried by the user. Image data fields are identified by the tag '*Image'* prefixed to an authentication token when specifying input data by the user.

4.4.6. API Web Services

WikiSensing supports several *API* web services that can be used by external platforms to automatically connect sensor devices to the system. These services include the functionalities to register users and sensors with the system as well as the storage and querying of sensor measurements. The key advantage of the service layer is the interoperability that enables anyone to use their preferred programming languages to connect with the system.

The *API* services in WikiSensing are implemented using *SOAP* (*Simple Object Access protocol*) [93] as well as the *REST* (*Representational State Transfer*) [94] protocols. While the *SOAP* services are for internal use and for testing and evaluation purposes, the *REST* services are exposed to the public via the WikiSensing web site. The accessibility, performance, scalability and support of multiple data formats such as *XML* and *JSON* is the main motivation for using the *REST* services over *SOAP* to interface the public usage [95].

The *API* web services reside in the *API Web Services* component in the application server of WikiSensing architecture and it uses the business rules and algorithms of the *Data Management* and *Data Aggregation* components (Figure 3.1). To access the *SOAP* web services a reference to the *API* needs to be obtained. Once this is done all service functionalities can be programmatically invoked. The following example code snippet written in *C#* illustrates obtaining a WikiSensing *SOAP* service reference. Subsequently the services can be accessed using this reference (e.g.*ClientWebreference*).

```
WikiSensingServiceReference.WikiSensingAPISoapClient
ClientWebreference = new
TestWebService.WikiSensingServiceReference.WikiSensingAPISoapClient();
```

The WikiSensing *REST* service *API* is implemented using *.Net C# 4.0* technologies and is accessible at WikiSensing.org. These services include *GET*, *POST* and *DELETE* functionalities to query, insert and remove sensor data. The system supports *XML* and *JSON* to send and receive data. These services are

executed via the *HTTP* protocol and are programmatically accessed using programming language such as *Java, C#, Python,* etc.

```
HttpWebRequest req = WebRequest.Create(uri) as HttpWebRequest;
HttpWebResponse resp = req.GetResponse() as HttpWebResponse;
```

The code snippet above demonstrates the use of the *.Net HttpWebRequest* and *HttpWebResponse* classes to obtain the functionalities of the provided services. The following code segment illustrates an example of posting the data and obtaining a response from the *REST* services. It loads the data that need to be submitted into a byte buffer. The length and the content type (e.g. XML or JSON) of the data are also specified. The request is then posted to the server and a response on the success of the *HTTP* post is finally obtained.

```
HttpWebRequest req = WebRequest.Create(uri) as HttpWebRequest;
byte[] buffer = Encoding.ASCII.GetBytes(content);
req.ContentLength = buffer.Length;
req.ContentType = "text/xml"; //OR req.ContentType = "text/json";
Stream PostData = req.GetRequestStream();
PostData.Write(buffer, 0, buffer.Length);
HttpWebResponse resp = req.GetResponse() as HttpWebResponse;
```

The following diagram (Figure 4.10) illustrates the sequential interactions between components that are involved in invoking API web services. The *HTTP GET* or *POST* request that is sent via the *API Web Services* component is relayed to the *Application Logic Framework* in the Application Layer. Subsequently the request is processed (e.g. input data mapped to object instances) and passed to the *Data Management* and then the *Data Access* component that queries or submits to the database. The output (result set for a query request or a success or failure notification for a submit data request) from the database is then relayed back to the client as an *HTTP* response.

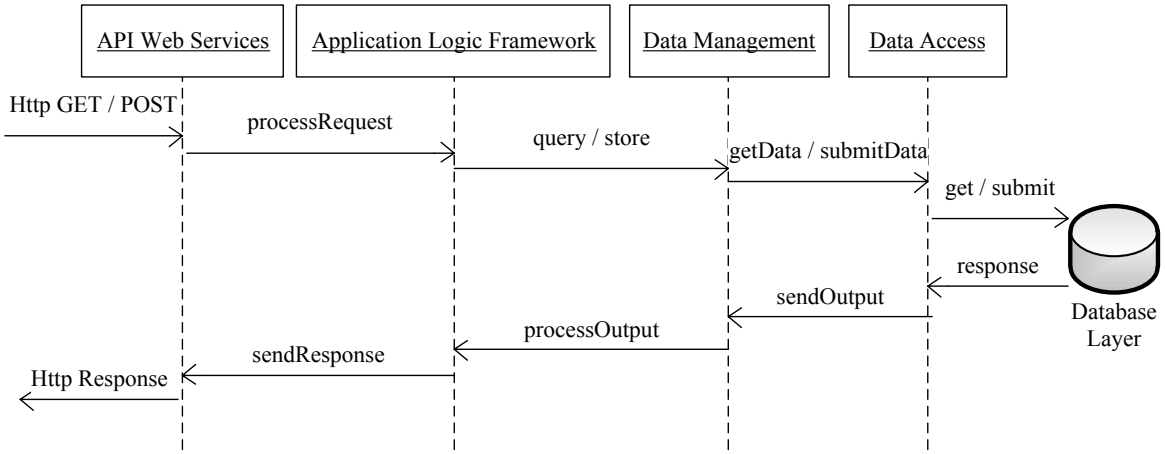


Figure 4.10: API Web Service sequence diagram

4.5. Evaluation

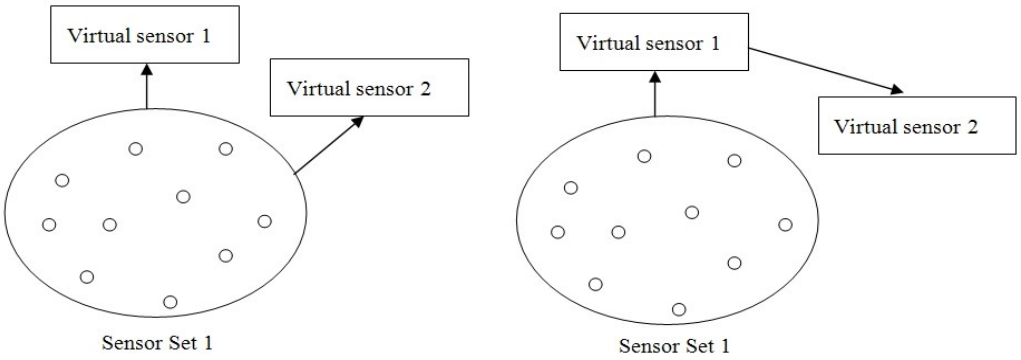
The experimental evaluation is designed to understand the attributes that affect the performance of the virtual sensors. The evaluation is based on different strategies that can be followed for aggregation queries and the storage for virtual sensor readings. The goal is to have an efficient methodology leading towards quicker responses to end users.

4.5.1. Improving the Performance of Aggregate Queries

Two scenarios are presented to demonstrate the methodology used by WikiSensing to improve the performance of aggregate queries for virtual sensors. The performance is based on the response time of these queries. Moreover the improvement of the response time is a reflection of the decrease in the number of database reads. Hence the aim is to identify strategies that can reduce the number of database reads. A virtual sensor is an aggregation of one or more sensor data streams. The aggregate function takes a set of data streams and produces a single value that summarizes the information contained in those selected data streams [96]. In the case of virtual sensors that are persistent, it records the results of the aggregation in the database.

Scenario 1: Aggregate sensor data streams to create virtual sensors that fully overlap with other virtual sensors.

Consider a scenario where a virtual sensor is already created using a set of sensors (virtual sensor 1, in Figure 4.11.a). A naïve strategy and the WikiSensing methodologies are discussed when the requirement for a second virtual sensor (virtual sensor 2) arises. Firstly a naïve strategy creates the new virtual sensor by including all the required contributing data streams in the aggregate query (virtual sensor 2, in Figure 4.11.a). This does not consider the fact that the fully overlapping virtual sensor 1 is a complete subset of virtual sensor 2. In contrast WikiSensing takes this overlapping of data into account and creates the virtual sensor 2 by using the information in virtual sensor 1 (Figure 4.11.b). It is assumed that virtual sensor 1 is persistent and continues to provide sensor measurements with its contributing sensors being active.



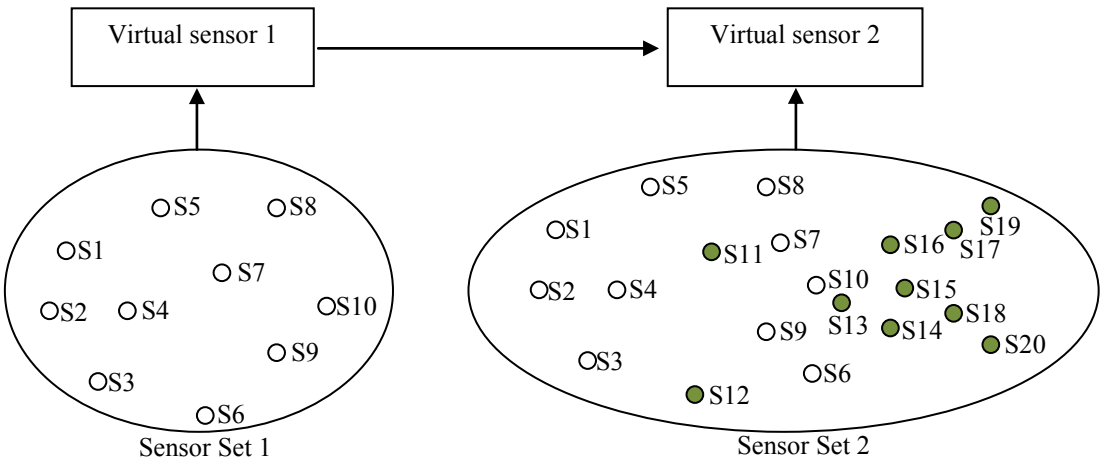
(a) (b)

Figure 4.11: Aggregate sensor data streams to create virtual sensors that fully overlap with other virtual sensors (a) in a naïve approach (b) in WikiSensing

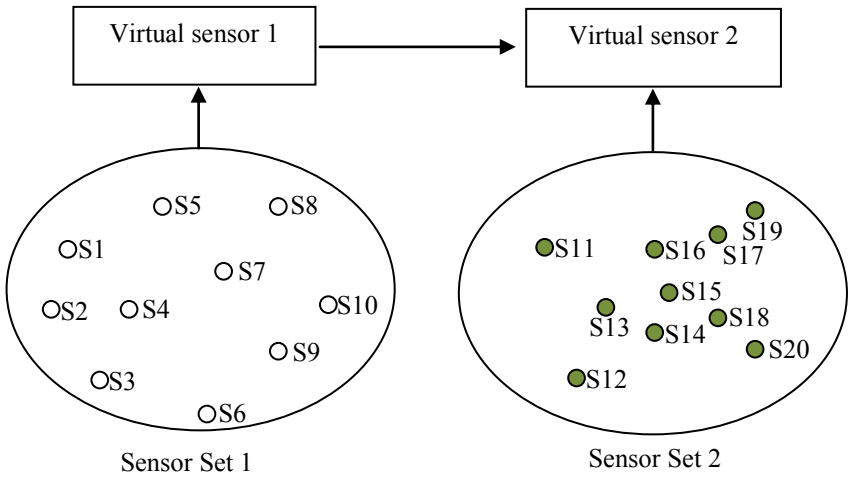
As the information of virtual sensor 1 is persistent and cached [97] the time involved in obtaining the result is expected to be less than a single database read. The aim of this strategy is to use existing persistent virtual sensors that are subsets of the newly created virtual sensor, in order to reduce the number of data base reads. The trade-off using this strategy is the extra cost of storing the sensor readings. Hence it is important to identify the situations where persistent storage is suitable (e.g. highly utilised virtual sensors).

Scenario 2: Aggregate sensor data streams to create virtual sensors that do not fully overlap with other virtual sensors.

Figure 4.12 depicts the requirement of a new sensor when the contributing streams do not fully overlap an existing virtual sensor (virtual sensor1). While a naive strategy would create new virtual sensor with all contributing sensors from scratch, WikiSensing uses the existing virtual sensor 1 and combines it with the other exclusive sensor streams. Similar to the first scenario, the readings of virtual sensor 1 can be taken from the cache and the rest of the reading can be fetched from the database.



(a)



(b)

Figure 4.12: Aggregate sensor data streams to create virtual sensors that do not fully overlap with other virtual sensors (a) in a naïve approach (b) in WikiSensing methodology

4.5.2. Experimental Setup and Benchmark

The version of WikiSensing used for the experiment is implemented as a complete working system hosted on an *IIS* server running on a Windows server 2008 virtual machine in the *IC-Cloud* platform [85]. The test emulator that implements the *Siege Benchmark* [98, 99] is used to send requests and runs in another *Linux Centos 5.4* virtual machine in the *IC-Cloud*. *Siege* is a regression testing and benchmarking utility that measures the performance of web applications and services.

The workload of the application tested obtains readings from physical sensors and virtual sensors that were created from a set of sensor data streams. The test emulator is run for a specific period of time and continuously generates a sequence of interactions that are initiated by multiple active sessions. After an interaction is completed, the emulator waits for a random interval before initiating the next interaction to simulate user's thinking time. Each experimental trial session is carried out for 300 seconds and three separate experiments are carried out. The performance is tested by obtaining random readings from sensor data streams.

The first experiment measures the response times of a physical sensor by increasing the number of users accessing it. Window sizes of 10 and 1,000 are used for a maximum of 1,000 simulated users. The second experiment involves a single client accessing virtual sensor readings. This is further divided into 2 trials which are tested with window sizes of 10 and 1,000 sensor readings. Each trial is tested with different workloads that are the naïve approach and the WikiSensing strategies based on a 100%, 80%, 50% and 20% overlap of sensors. The third experiment has the same parameters as the previous one, except that it is tested using multiple simulated users with active sessions. The first trial simulates 100 clients concurrently accessing the system with the gradual increase of the contributing sensors. The second trial gradually increases the number of clients that access a virtual sensor created with 50 sensor data streams.

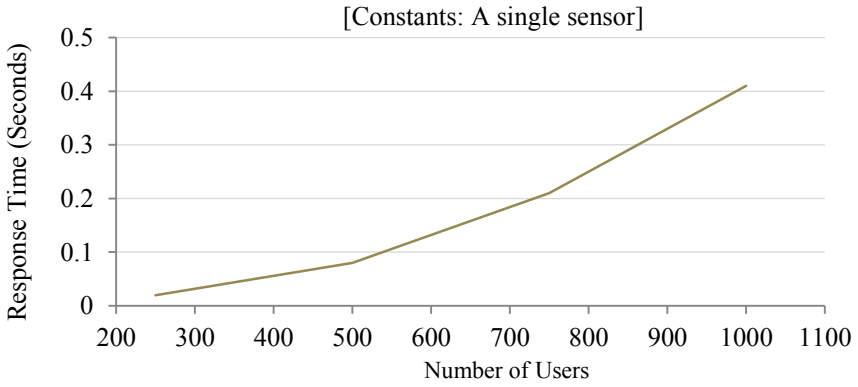
The test emulator based on the *Siege Benchmark* outputs the response time for each experimental scenario. The emulator makes an *HTTP* request for a web page that invokes a web service function. The response time is calculated from the start of the invocation till the function returns a value and is loaded into the web page. The time for each execution is summed and averaged to obtain uniform reading. Table 4.6 summarises the setup used for the experiments.

Experiment	Constant Parameters	Altered Parameters
1 (a)	Single sensor, measurement window size of 10	Number of concurrent clients increased
1 (b)	Single sensor, measurement window size of 1000	Number of concurrent clients increased
2 (a)	Single sensor, single client, measurement window size of 10	Number of contributing sensors increased
2 (b)	Single sensor, single client, measurement window size of 1000	Number of contributing sensors increased
3 (a)	100 concurrent clients, measurement window size of 10	Number of contributing sensors increased
3 (b)	50 sensors, measurement window size of 10	Number of concurrent clients increased

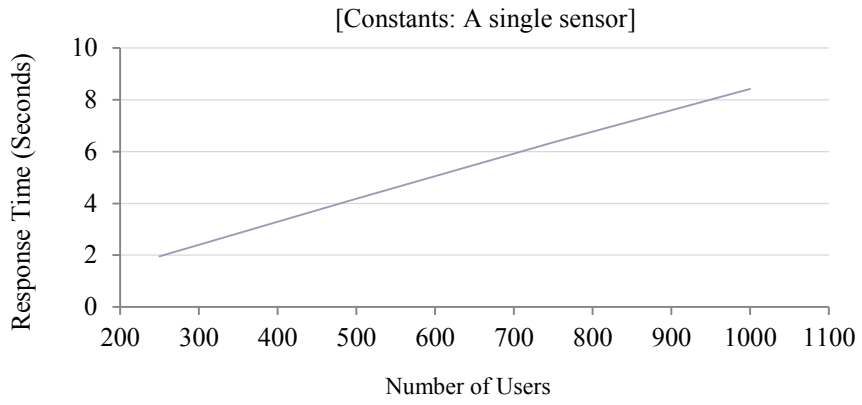
Table 4.6: Summary of experimental setup

Experiment 1: Measure response time of a physical sensor accessed by an increasing number of clients

The response time of obtaining readings from a physical sensor is tested with the increase of the number of users. This results in increasing the number of concurrent users that access a single sensor stream with window sizes of 10 and 1,000.



(a)



(b)

Figure 4.13: Response times for querying a single physical sensor by increasing the number of clients (a) Window size of 10 (b) window size of 1000

The number of concurrent clients is increased from 250 to 1,000. The response time *R(t)* has a dependency on the number of concurrent users (*X*) and the window size *(Y), R(t) = f(X,Y)* according to the graph (Figure 4.13).

Experiment 2: Measuring response time of virtual sensors accessed by a single client with respect to the increase of the contributing sensor data streams.

The response time for obtaining an aggregate reading from a virtual sensor is measured with respect to the increase of the number of contributing sensors. The aggregate reading is a combined (e.g. average) value of the contributing sequential data streams. It tests a single client accessing the virtual sensors reading by gradually increasing the number of contributing sensors from 10 to 140. The different workloads are the naïve approach where all records are fetched from the database, 100% overlapping where the information is picked from the server cache and 80% 50% and 20% overlapping where the data is fetched directly from the database.

Virtual sensor readings are cached when the user makes a request for that sensor. If the data is not cached it is then fetched from the database. Overlapping is dealt with in WikiSensing as illustrated in Figure 4.12.b. For example, if the overlapping is 80% for a virtual sensor it obtains the overlapped portion using a single database read (or directly from the cache if the information is cached) and gets the rest (20%) of the reading from the other data streams.

Two trials are used with windows sizes 10 (Figure 4.14 (a)) and 1,000 (Figure 4.14 (b)). The aim of changing the window size is to alternate the amount of sensor readings that are selected for an aggregate query. For instance, a window size of 10 selects the 10 most up-to-date sensor readings for the aggregate query.

The response times for both the scenarios with a 100% overlap (fetched from the database and the cache) were constant throughout the experiment and returned response times of 30 and 10 milliseconds. With a window size 10, the response time of a single virtual sensor is in the range of 60 to 20 milliseconds for the naïve, 80%, 50% and 20% overlapping workloads. The performance for a single virtual sensor when used with window size of 1,000 is in the time span of 110 to 30 milliseconds for the respective workloads.

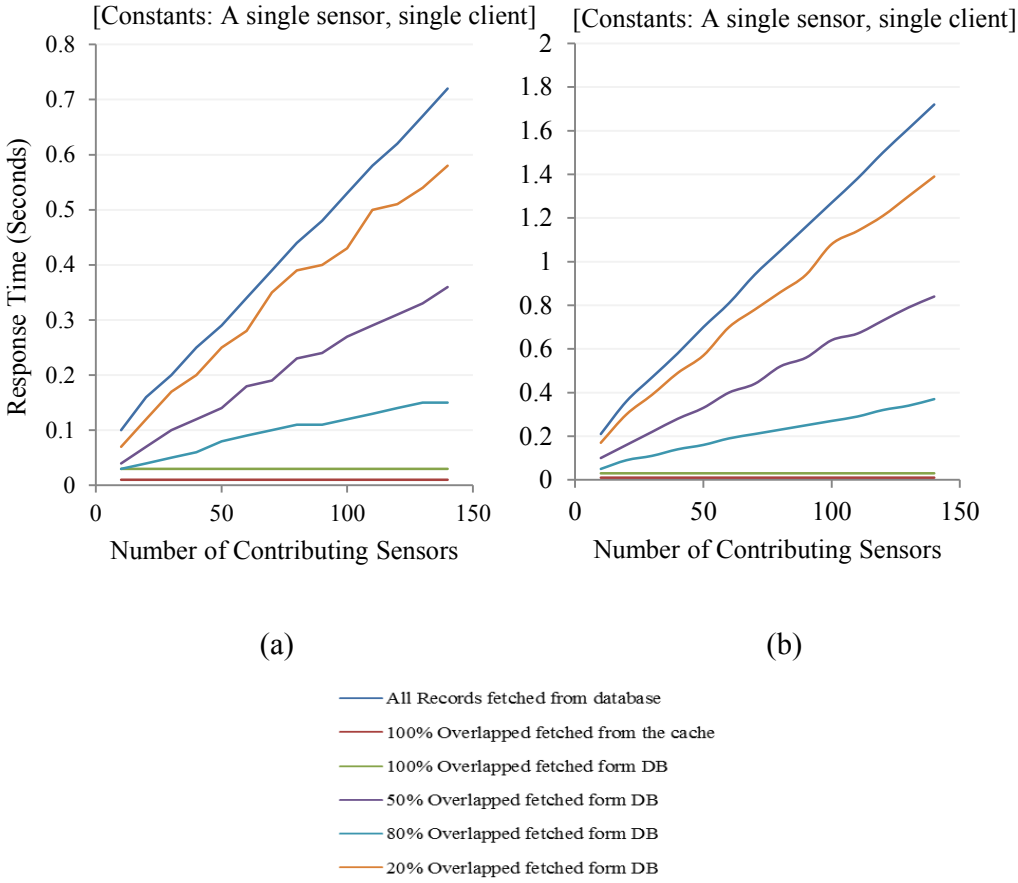


Figure 4.14: Comparing the response times for querying a single virtual sensor with (a) window size of 10 (b) window size of 1,000

The response time for the virtual sensors readings *R(t)* has a dependency on the number of contributing sensors (*X*) and the window size *(Y), R(t) = f(X,Y)*. When comparing the results of the two window sizes the different strategies have responded in similar fashion. The main difference here is that the response time increases when using a window size of 1,000. The response time of the 50% overlapped workload at 140 sensors (window size 10) is 370 milliseconds. This response time increases when the overlapping is less and reduces when the overlapping is high. This is due to the impact of the increase in the number of database reads. Thus the decrease of overlapped sensors constitutes a 60% change of the response time. The same situation prevails with a window size of 1,000 as well.

Experiment 3: Measuring response time of virtual sensors (a) accessed by 100 concurrent clients by increasing the number of contributing sensor data streams, (b) containing 50 sensors by increasing the number of concurrent clients

This test simulates a case where a popular (high usage) virtual sensor is accessed by many users. In the first trial the response time of a virtual sensor is measured with 100 clients accessing the same set of data concurrently. The second trial records the response time by increasing the number of clients from 10 to 50 and keeping the number of contributing sensor data streams constant at 50. In both trials we use a window size of 10. This experiment mainly focuses on testing the response time and the scalability of the system. The graphs in Figure 4.15 depict the bottlenecks with the scenarios when fetching data when the overlap does not exceed 50%. The scenarios with 100% overlapping fetched from the database and memory cache returned constant response times ranging from 30 and 10 milliseconds throughout this experiment.

The test emulator times-out due to memory limitation when using a traditional naïve strategy when the number of sensors exceed 50 as depicted by the graph in Figure 4.15.a. Clearly the strategy followed by WikiSensing to use overlapping resulted in comparatively less response times than traditional approaches and hence offers better scalability.

The response time for the virtual sensors readings *R(t)* has a dependency on both the number of contributing sensors (*X*) the window size (*Y*) and the number of concurrent users *(Z), R(t) = f(X,Y,Z)*. As the data access intensifies with 100 concurrent users the response time tends to increase and the performance is diminished in the strategies where there is 50% or less overlapping. From these experiments it can be concluded that the response time for virtual sensor readings for the naïve strategy (formula 4.1), when information is cached (formula 4.2) and when data is fetched from the database (formula 4.3) are:

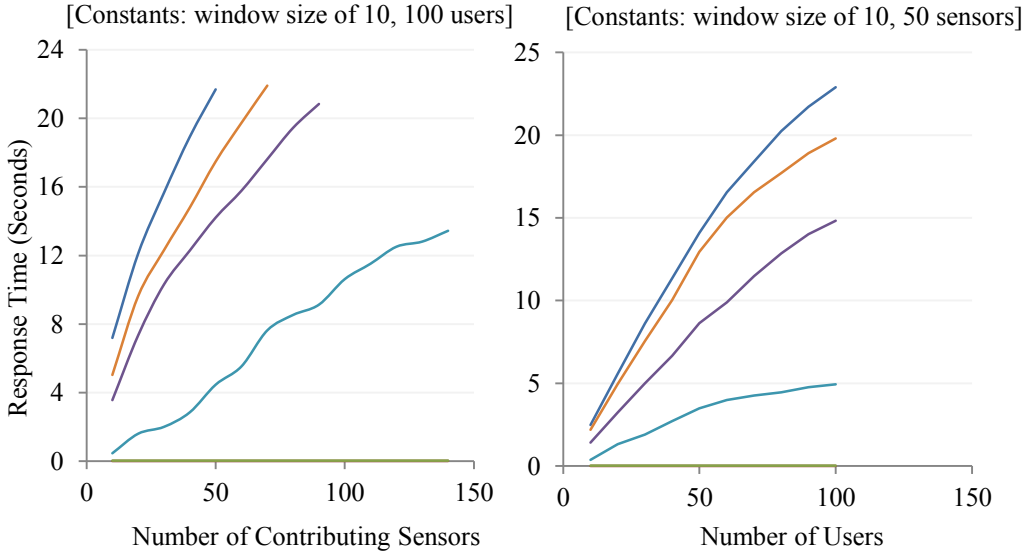
$$R(t)\ (Naïve) = N * d(t) + a(t) \quad (4.1)$$

$$R(t)\ (WikiSensing, when\ cached) = c(t) + (N - O)\, d(t) + a(t) \quad (4.2)$$

$$R(t)(WikiSensing, when\ fetched\ from\ database) =$$

$$d(t) + (N - O)\, d(t) + a(t) \quad (4.3)$$

N denotes the number of contributing sensors in the virtual sensor and *O* denotes the number overlapped sensors. The time intervals involved in the access strategies are the time to fetch records from database (*d(t)*), the time to fetch records from cache (*c(t)*) and the time to process the aggregation (*a(t)*).



(a) (b)

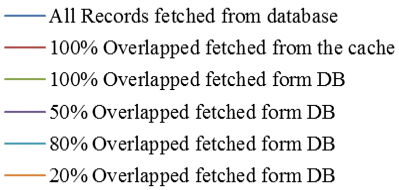


Figure 4.15: Response times for querying a single virtual sensor increasing the number of (a) contributing sensors with 100 concurrent users (b) users with 50 sensors

The other factors that affect the response time of such an *HTTP* request are the performance of the browser, the speed of the Internet connection, the local network traffic, the load on the remote host, and the structure and format of the web page requested [100]. Taking the time cost of all these factors as *X*, the total response time is = *R (t)* + *X*.

4.6. Related Work

Sensor data management systems contain large amounts of data sets and a high throughput of access to this information can challenge the capacity of a single server. While high query rates can slow down performance of the server, the increase in demand for storage can exceed the capacity of a single machine. A key design factor of WikiSensing is the adaptation of the non-relational *MongoDB*. The *Sharding* approach used by *MongoDB* based on horizontal can be compared with another popular method known as vertical scaling of data.

The vertical scaling [101] approach adds more processing power and storage resources to increase capacity. The problem with this strategy is that in cloud-based systems like WikiSensing, the cloud providers only allow users to provision smaller instances of virtual machines or computing power with a maximum capability for vertical scaling. The approach used for scaling in WikiSensing is the *Sharding* mechanism by *MongoDB* that horizontally scales the data sets by dividing and distributing it over multiple servers (*Shards*). The *Shards*

collectively make up a single logical database each shard is an autonomous database. This process reduces the number of operations each machine handles as the load is distributed when more *Shards* are introduced which will increase capacity and throughput horizontally.

The service *API* in WikiSensing can be compared with the features supported on the *Xively* [9] sensor data management system. However the storage and querying of heterogeneous data supported by WikiSensing is not available on *Xively*. They are fixed to single schema for the sensor details and the stream information. This is also a motivating factor to develop a sensor data management system so that various formats of data can be stored and analysed. This is further exemplified in the following chapters where trustworthiness is managed in different dimensions of sensor data.

4.7. Conclusion

This chapter presented the implementation details of WikiSensing and described a set of case studies to demonstrate some of its functionalities. The implementation of the hybrid data storage, the online collaboration, the *API* service layer and the feature of creating virtual sensors are highlights in this discussion. The hybrid data storage is designed to store sensor data with different characteristics (continuous data as opposed to intermittent data) using different storage strategies. Online collaboration in WikiSensing is supported using a wiki framework, allowing users to provide their feedback or comment on the sensor data. Interoperability is achieved in WikiSensing by providing an *API* service layer implemented using the *REST* and *SOAP* web service protocols. Virtual sensors are a novel feature introduced here to obtain measurements when sensor data are insufficient, or when a direct sensor measurement at a specific location is missing. The system is evaluated on the performance based on the response time of queries on these virtual sensors.

An important future development would be to trace the modifications of virtual sensors. Hence there are plans to extend the data model in order to maintain a record of changes applied to virtual sensors. A potential source for this information could be the updates applied to the virtual sensor network and the virtual sensors query entities. The work done by [102] highlights the challenges in managing historical sensor information and can be used as the basis for this development. Additionally, there is scope to further standardise and improve the WikiSensing query constructs. It is possible to use the *BNF* (*Backus Normal Form*) [103] notation technique for context-free grammars to describe the syntax of the query language. The *BNF* grammar and semantics can also be used to define the actions (e.g. inputs) of these query constructs.

5. Modelling and Managing Trustworthiness

In this information age, vast amounts of data and knowledge are unevenly dispersed around the world. Online collaboration has facilitated the convergence of knowledge and made information more accessible to everyone. Online shared data is becoming ever so popular with the increase of usage in online collaborative systems such as *Wikipedia*, *OpenStreetMap*, *Xively*, etc. They have become the basis of knowledge sharing among users with various experience and backgrounds around the world. People tend to learn, refer and obtain up-to-date information from these sources. The reason for the success of these online collaborative systems is that the internet has made such resource-sharing quicker, easier and cheaper.

The open nature of collaborative systems enables interested users to update and add information. The openness is clearly an important aspect in the success of collaborative systems. However it also incurs the problem on the lack of trustworthiness of the shared knowledge and sources of information. Hence the focus of this chapter is to model and manage the trustworthiness of such collaborative data. The domain of interest is based on sensor data that is collected in WikiSensing. Ideally what is needed is an indication e.g. an assessment or a rating on the trustworthiness of the shared sensor data. This can be helpful for the users to make a judgement on whether to accept or reject the information.

This chapter describes the development of a framework and methodology for trust management in collaborative sensor systems. A Bayesian definition of trust is used in this methodology, with metrics being used to model different types of available evidence. The evaluation of this approach is based on a case study in environmental modelling over pollution data.

5.1. The Requirements (Challenges)

The increase in the use of sensors and sensor networks [6] to measure and collect information from physical environments has recently given rise to the development and use of collaborative sensor management systems [7-11]. With such systems, users can collaborate on the collection and analysis of environmental data from different locations as well as use such data to build new applications. A key challenge in such systems, however, relates to the trustworthiness of data itself. The data is collected from sensors owned by third parties and not under the user's control. Individual sensors could be reporting untrustworthy or wrong values for many reasons. They could be faulty, mis-calibrated, beyond their life time or could have stopped working completely. They could also have been hijacked by malicious attackers and forced to report wrong measurements. The aim of this chapter is to investigate how such issues can be addressed by building a generic framework for modelling and evaluating sensor trustworthiness.

To date, little work has been conducted in developing a generic trust modelling framework for collaborative sensor systems. Moreover, there is currently no standard, or agreed upon definition for the concept of sensor data trustworthiness that can be used generically. There is also little work defining what information needs to be collected about the sensors, or their measurements, for use in a generic trust modelling framework. The aim of this chapter is to investigate how to address these issues with a view to allowing users themselves to model and evaluate sensor data trustworthiness based on the evidence that may be available to them about the sensors and their measurements.

5.2. The Definition

This thesis builds on, and extends, the general framework for defining trust provided by *Sun et al.* [63]. In their work, they define trust as a relationship established between two entities for a specific action. One entity, called a *Subject*,

trusts the second entity, called an *Agent*, to perform an *action*. The concept of trust in this framework describes the subject's view of whether the agent will perform the action. The generic trust relationship can be defined using the notation *{Subject: Agent.action},* and *P{Subject: Agent.action}*denotes the probability that the agent will perform the action in the subject's point of view. The advantage of this approach is that this probability is not absolute, but reflects the opinion of a specific subject. Thus, different subjects can assign different probability values for the same agent and the same action. It is noted that the probabilistic models used by *Sun et al.* as well as others [63] are based only on whether a series of historical interactions between the user and the sensor, i.e. measurements provided by the sensor to the user, were acceptable to be correct or not.

In *Sun et al.*'s work, both the subject and agent traditionally represent sensors in a fully autonomous sensor network. In this case, sensor nodes exchange information and have to decide, based on historical values only, which sensors are reliable and which are not. This naturally leads them to use a binomial distribution model based on the user's observations of the sensor's previous measurements. In contrast, this work uses the same conceptual framework, but considers the *Subject* to be the user (human being) of the collaborative data management system and the *Agent* to be a specific sensor registered in the system and the *action* is a specific measurement. This research aims to develop a more generic approach for modelling trust that considers other forms of evidence (*E*) available to the user, not only the list of historical actions.

Let *T {User: Sensor.measurements, E}* denote the trust value of the relationship between the user and the sensor measurements and let *P {User: Sensor.measurements, E}* denote the probability that the sensor provides measurements that are accepted by the user. In this definition, *E* represents evidence or additional information that can be used to assess trustworthiness.

Evidence (*E*) can include historical information (*H*) on the interactions between the user and the sensors. It can also include evidence on conflicts (*C*) between the sensor measurements with those of other sensors, as well as conflicts

with background information known to the user. It can include contextual information (*X*) about what the sensors are measuring and where they are deployed. It can also include subjective information provided by the views of other users (*V*) on either the sensor or on its particular measurement. The evidence set $\{C, H, V, X, \ldots\} \in E$ is extensible as needed. Furthermore, as shown in Figure 5.1, it also allows evidence to be organized and modelled hierarchically if needed. For example, conflict information (*C*) can naturally be divided as conflict with other sensors (*O*) and conflicts with background information (*B*). The contextual factors (*X*) can be modelled as different factors F_1, F_2, …. etc., capturing information on such issues such as the calibration, exposure, as well as any factors that influence the sensor readings in general.

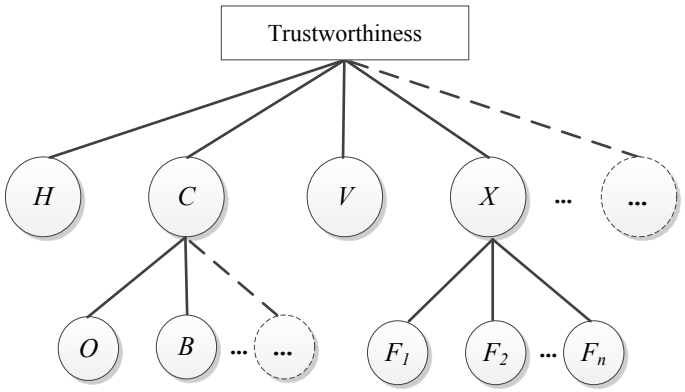


Figure 5.1: The model for trustworthiness metrics

Table 5.1 lists the attributes that influence the trustworthiness along with other symbols used in this chapter.

Symbol	*Description*
T	The trustworthiness of the sensor measurement
C	Conflicting information
H	Historical information metric
V	Views of experts metric
X	Contextual factors
O	Conflicts with other sensors metric
B	Conflicts with background information metric

Symbol	*Description*
E	The attributes (Evidence)
$F_1 \ldots F_n$	The set of contextual factors
M	Sensor measurement
W	Window size of measurements
x_i	Sensor properties

Table 5.1: Description of Symbols

5.3. Bayesian Modelling for Trustworthiness

A Bayesian probabilistic approach is followed for modelling Trust, T, as $P(T|E = e)$, where T is the hypothesis, such that a sensor is trustworthy, given the observed set of measurements $E = e$. Without loss of generality T can be regarded as a binary variable (trustworthy or not-trustworthy). Given historical data, it is possible to train binary Bayesian classifies [104] to predict the class membership probabilities, i.e. to determine the probability that a sensor measurement is trustworthy or not. The approach requires defining metrics to measure and represent the different forms of evidence available and requires collecting a training data set to calculate the required statistics. The following sections describe the examples of metrics and how they can be collected. The remainder of this chapter describes the Bayesian modelling approaches used.

Consider the event of the sensor measurement being trustworthy as $T_1, T_2, \ldots, T_n$. The probabilities $P(T_1), P(T_2), \ldots, P(T_n)$ are the prior probabilities of the events that determine the trustworthiness of the sensor measurement. $P(T_i)$ is the probability that T_i is correct. It is assumed that the collected metrics that act as evidence gives the information on the correctness of the hypothesis. Computing $(T_i|E)$ is required, as shown by the following formula (5.1)

$$P(T_i|E) = \frac{P(E|T_i)P(T_i)}{P(E)} = \frac{P(E|T_i)P(T_i)}{\sum_{k=1}^{n} P(E|T_k)P(T_k)} \quad (5.1)$$

T is the hypothesis that a sensor is trustworthy and *E* represents the evidence that is constituted using the provided metrics e.g. *E = C {O, B}, H, X {F1, F2, ..., Fn}, V*. *T* has two different classes (Trustworthy and Not-Trustworthy).

5.3.1. The Naïve Bayesian Model

A Naïve Bayesian classifier selects the most likely classification of T_i (the trustworthiness) given the metric values m_1, m_2, ..., m_n. The probabilities of $P(e_1|T_1)$, $P(e_2|T_2)$, ..., $P(e_n|T_n)$ are estimated from the training data. The assumption of class conditional independence is made to indicate that there are no dependence relationships among the metrics. We can also use continuous value representation of the metrics when training the classifier. In this case, the values of metrics can be assumed to have a *Gaussian* distribution (g) where μ_{T_i} and σ_{T_i} are the mean and the standard deviation of metric e_k(formula 5.2).

$$P(e_k|T_i) = g\left(e_k, \mu_{T_i}, \sigma_{T_i}\right) \quad (5.2)$$

5.3.2. The Bayesian Network Model

The key assumption of the Naïve Bayesian classifier is that the metrics are independent. If the metrics are not independent, then a Bayesian Network can be used to model conditional dependencies among them. In this case, the joint probability metrics $(m_1, \ldots, m_n)$ of the evidence can be computed using formula (5.3). The values $P(e_i|Parents\ (E_i))$ correspond to the records in the conditional probability tables for E_i in the Bayesian Network.

$$P(e_1, \ldots, e_n) = \prod_{i=1}^{n} P(e_i|Parents\ (E_i)) \quad (5.3)$$

Figure 5.2 illustrates a Bayesian Network model designed by a domain expert to represent the metrics and their dependencies for this research. An arrow indicates a dependency and the circlers represent the metrics or attributes. F_1 to F_3 are contextual information. For example, in the case of measuring pollution values

in an urban area, F_1 may represent data on a nearby pollution source (e.g. factories near each sensor), F_2 can represent information on sensor exposure (e.g. whether it is deployed indoor or outdoor), F_3 can represent information on sensor calibration. Noteworthy dependencies exist between these contextual factors and other calculated metrics (e.g. *H, B, O* and *V*) as well as the actual trustworthiness. For instance the sensor not being calibrated (F_3) can make its measurements conflict with other sensors with background data, and affect user ratings. Moreover, the sensor exposure (F_2) can have a temporal impact on its measurements that usually affect the history of readings. For example, a sensor placed outdoors may sometime produce wrong measurements when it rains. In contrast influencing factors (F_1) such as a nearby factory can impose conditional dependencies on the trustworthiness of the sensor. For example, although the sensor trustworthiness is usually diminished when it conflicts with its neighbouring sensors, the existence of an influencing factor may provide the rationale for explaining such conflict.

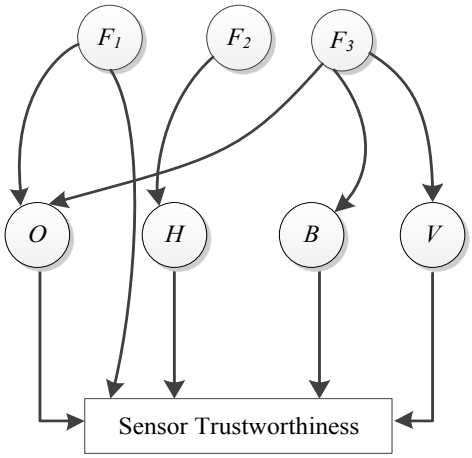


Figure 5.2: A Bayesian Network trust representation designed by domain expert

Mengshoel et al. [62] uses Bayesian Network approach to detect faults with sensors in a aerospace vehicle. Bayesian modelling is used as it provides a basis for reasoning on sensor faults and also to probabilistically determine the health of a hardware component in an aerospace vehicle. Their approach uses actual activities involved with such vehicles to be represented as noted in the Bayesian Network. The work described in this research uses a similar approach of

Bayesian modelling. However the Nodes of the Bayesian Network are metrics generated from sensor data as well as contextual factors. Moreover the same metrics are used in a Naive Bayesian approach for comparison as well as extending the list of metrics as needed.

The causality relationships in the Bayesian Network were derived based on the knowledge and experience of an expert in this particular domain (sensor types and its geography of deployment). Hence it must be noted that different experts may also have different opinions on how the dependencies are set. This is especially important when more metrics and contextual data are involved. Moreover to address this issue it would require the use of either a dynamic modelling approach in order to identify the most suitable of models.

5.4. The Methodology and Implementation

The methodology has three main stages (Figure 5.3). The first stage involves the collection of sensor data (measurements, sensor properties, etc.), user feedback and values for input parameters for the trustworthiness models. The second stage manages the collected data and utilizes the trustworthiness models and formulae to calculate the metrics and trustworthiness values. The third stage outputs calculated trustworthiness and supporting information.

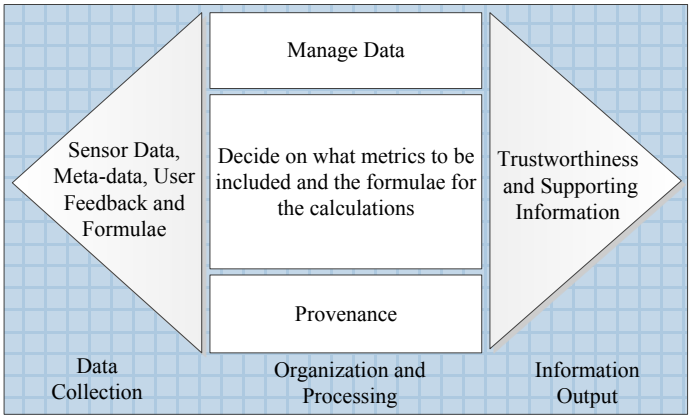


Figure 5.3: The Sensor Trustworthiness Management Process

During the first stage the required data for the trustworthiness assessment is collected. This data comprises of sensor data, user feedback and the input parameters. The sensor data includes sensor measurements, meta-data (sensor properties) and geographical details of the deployment of the sensors. The user feedback contains user ratings or any type of positive or negative remarks. The input parameters are used by the formulae that calculate the trust metrics that are discussed in detail in section 5.5.3.

The second stage manages the unstructured data and utilizes the formulae to calculate the metrics. The user can also provide additional metrics and formulae for their calculations as well as their own formulae for the existing metrics. Curating the data is a prerequisite for the metric calculations e.g. sampling different sensor reading frequencies and converting sensor measurements into a common unit of measure. The second phase also involves calculating the trustworthiness rating for the sensor. In order to calculate the trustworthiness of a sensor or sensor measurement a set of metrics are formulated. These metrics are representations of data that can include historical information (*H*), information on conflicts (*C*) between the sensor measurements with other sensors (*O*), conflicts with background information (*B*), contextual information (*X*) (e.g. calibration) and information provided by the views of other users (*V*). Information on these calculations is also stored for provenance. Further this process can be reinitiated on the same sensor at a later time or when new information becomes available. This model highlights the importance of data provenance as the trustworthiness of a sensor may change over time as well as when new information becomes available.

The third phase outputs the calculated trustworthiness and all supporting information. This supporting information is used to explain the calculations and the parameters used in the calculation of the metrics and the final trustworthiness rating.

5.4.1. The Architecture

The WikiSensing architecture described in section 3.2 is extended in order to support this generic framework by introducing new components highlighted in bold in Figure 5.4. It is noted that the components themselves are implemented in a generic way and can be accessibly plugged into a sensor data management system other than WikiSensing.

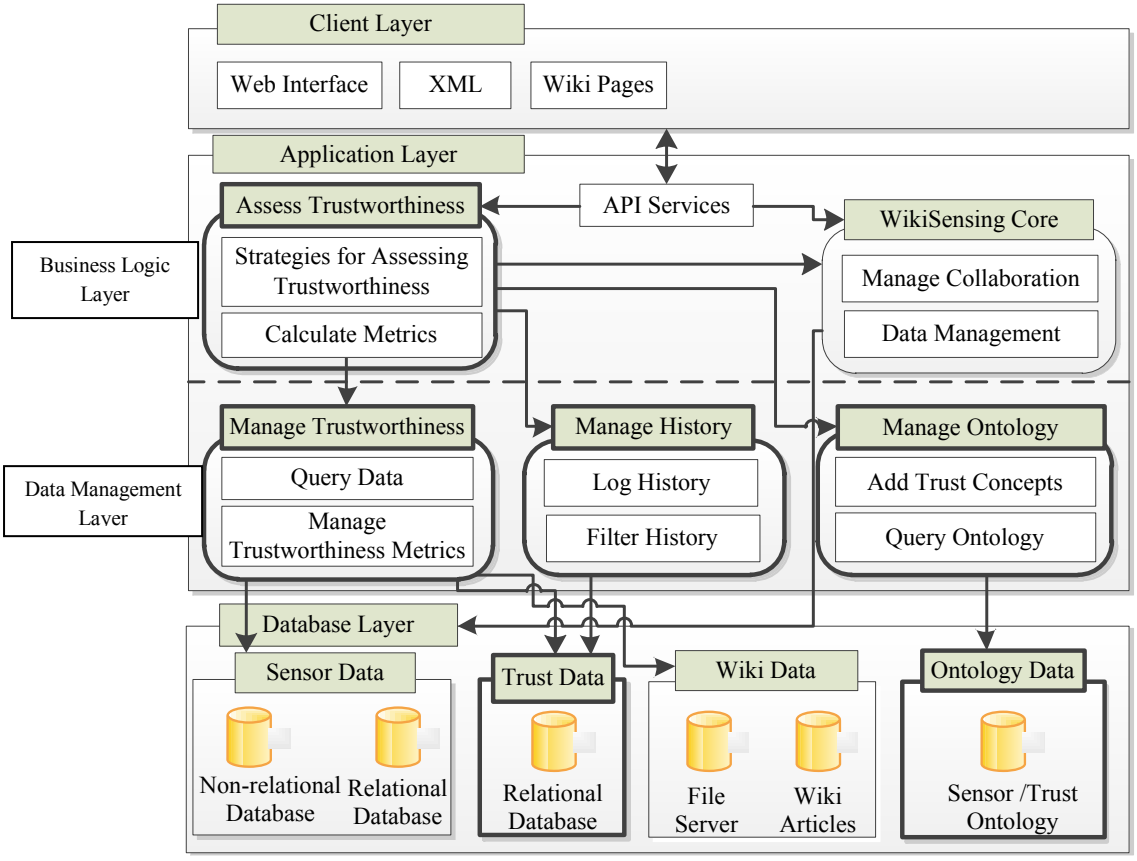


Figure 5.4: The architecture of the trustworthiness management framework

The overall architecture is based on a layered model with a data layer that includes a database for trustworthiness data. The algorithms for trustworthiness management reside in the application tier. Moreover the metric calculation is done in the *Assess Trustworthiness* component and is invoked by *API services*. The

metric calculation requires the functionalities of the *Manage Trustworthiness*, *Manage History, Manage Ontology* as well as the *WikiSensing Core* components.

The database layer contains the databases for the sensor, trustworthiness, wiki and the ontological data. The client layer provides a web interface for sensor data management and Wiki pages for collaboration with *XML* being used as a medium for the exchange of data.

The application tier contains two sub layers, a business logic layer (top) and a data management layer (bottom) and components with thick borders are specifically responsible for trustworthiness management. The data management layer provides functionality for data manipulation and the business logic layer contains algorithms for resolving conflicts and assessing trustworthiness. The *Assess Trustworthiness* module uses the *WikiSensing Core* components and the data management layer to obtain information from the databases for metric calculations.

Once the metrics are calculated it is then represented as ontology and the calculations and data are stored in history for provenance. For instance, when a trustworthiness assessment request is made by the *API Services*, the *Assess Trustworthiness* module obtains the strategies (formulae) for the metric calculation. It then obtains the necessary data (sensor data, meta-data, user ratings, etc.) and calculates the trust metrics. All calculation details and metrics are logged using the *Manage History* module. The *Manage Ontology* then represents this information in the trustworthiness ontology as individuals based on the defined ontology schema. The metric calculations usually require data from the sensor database that includes current and historical measurements, spatial information (e.g. geographical coordinates) and sensor types. It also requires data on sensor properties and context (represented as ontology) as well as user rating information (recorded in wiki pages). All the current metric values as well as their historical values are stored in the trust database.

5.4.2. Representing Trustworthiness Metrics as an Ontology

The trustworthiness metrics, the contextual data and the sensor information are stored as ontology in order to maintain a common vocabulary. This research extends the *OntoSensor* ontology [66] to contain sensor trustworthiness data. *OntoSensor* is an extension of *SUMO* (*Suggested Upper Merged Ontology*) [65] a top-level ontology for computer based information systems that provides concepts that are general throughout the knowledge domain. The *OntoSensor* ontology is a comprehensive ontology that maps a subset of the *SensorML* [64] concepts into *OWL* [67]. The *WikiSensing* trustworthiness ontology is available on the internet under the section Trustworthiness *API* at wikisensing.org.

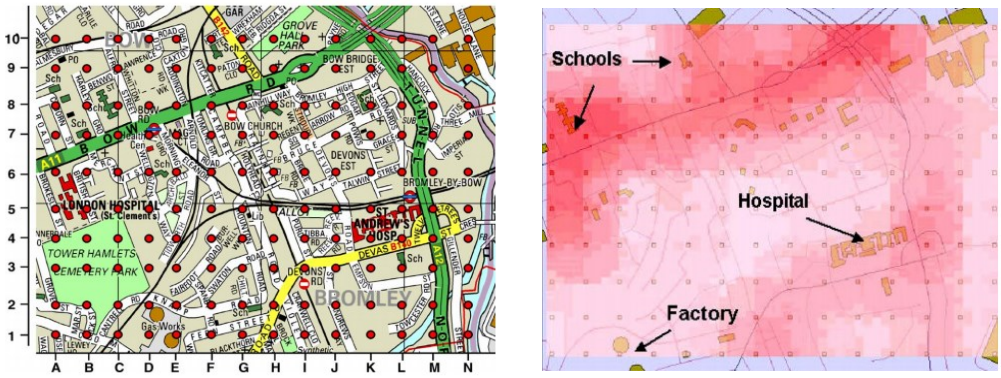
5.5. Example Scenario

An example scenario is used to demonstrate how trustworthiness is assessed in a specific domain of sensor data. The trust metrics are calculated for a data set collected from pollution sensors known as *GUSTO* sensor. The issues of assessing and measuring conflicts between sensors which are needed for the metric calculations are also discussed. This is followed by an example of a Bayesian Network model designed by domain expert for the metrics and the ontological representation of these metrics and trustworthiness of the sensors.

5.5.1. The *GUSTO* Data Set

The original *GUSTO* data set used for the case study is archived data (recorded in June 2003) that consists of pollutant readings and time stamps at a busy location in East London. The source of the data is *GUSTO* [22] (Generic Ultraviolet Sensors Technologies and Observations) sensors. It is based on open-path *DUVASTM* (Differential Ultraviolet Absorption Spectroscopy) technology and measures and transmits the volume mixing ratios of key urban open air path pollutants in real-time. The key distinguishing features of *GUSTO* sensors are its short time scale (of

order 2 s scan rate), open variable path (up to 30 m), enabling measurements to be carried out in situ and localized effects to be characterized and relatively cheap and robust, sufficient for large-scale deployment. The data contains readings of 140 sensor nodes that are deployed in a grid (Figure 5.5) with each sensor node containing four sensors measuring *NO, NO2, SO2* and *ozone* pollutant levels.



(a) (b)

Figure 5.5: GUSTO sensors (a) The deployment grid in East London (b) The annotation of sensor map

The data set is for a single day, reported at 1-minute intervals from 8:00am till 6:00pm (600 measurements per sensor and 2,400 measurements per sensor node) and captures the effects of traffic patterns on specific roads, as well the operation of other pollution sources (e.g. factory).

The *GUSTO* data set is used in this research as an example scenario to demonstrate how trustworthiness metrics are calculated and also to evaluate the methodology. The remainder of this section focuses on how the trustworthiness metrics can be calculated for *GUSTO* data set based on the characteristics of the sensors, their measurements, and the evidence available about them. The following sections describe the various experiments carried out for the metrics to develop and evaluate the trustworthiness models. In these experiments, the original data set is treated as one originating from trustworthy sensors, and other data sets are created that introduce specific errors in some of the sensors to simulate untrustworthy sensor behaviour.

5.5.2. Assessing and Measuring Conflicts

Assessing and measuring abnormal readings and conflicts play an important part in assessing trustworthiness of sensor measurements and are also used in this metric calculation approach. A problem arises when there are multiple sensor devices deployed at a particular location providing varying readings or seemingly conflicting measurements. It is important to establish whether these differing measurements are mere acceptable variations (e.g. due to sensor accuracy) or genuine conflicts. To address this issue, definitions are used based on the *JCGM* – VIM [105] standards for accuracy, precision and uncertainty of sensor measurements. These standards are used as it provides standardised and common terminologies across different fields of science.

The sensor *Accuracy* is the maximum difference that will exist between the actual value and the indicated value at the output of the sensor. Moreover *Precision* refers to the degree of reproducibility of a measurement where if exactly the same value were measured a number of times, an ideal sensor would output exactly the same value every time. *Uncertainty* of a measured value is an interval around that value such that any repetition of the measurement will produce a new result that lies within this interval. These properties are explicitly used in this methodology in defining a conflict as they layout variations that can exist in sensor measurements.

The aim is to use these properties in order to establish a value that can be used to compare two sensor measurements. Consider measurements m_1 and m_2 (where $m_1 \neq m_2$) from two sensors placed at the same location. We use these sensor properties to check if the disparity of measurement is a conflict or an acceptable variation. A measurement is conflicting when the differences between the readings of two sensors m_1 and m_2 is greater than a property value of x_i where x_i can be the accuracy, precision or uncertainty interval as shown in the following formula (5.7). If not satisfied, the measurements can be considered as acceptable or varying.

$$|m_1 - m_2| > x_i \; where \; \{x_1, \; x_2, \ldots \; x_n\} \; x_i \quad (5.7)$$

Having a measurement that is within acceptable varying range does not affect the trustworthiness rating of a sensor, but is impacted when measurements conflict. A weighting based on the distance between the sensors is used as a coefficient when comparing measurements. This definition is used throughout the chapter when identifying conflicting sensors.

5.5.3. Calculating the Metrics

To enable trust modelling, a set of trustworthiness metrics are proposed for the sensor measurements represented in Figure 5.1. Based on the properties of the *GUSTO* example metrics can be easily calculated. For historical information (*H*) abnormal measurements recorded previously by the sensor are considered. For example, in this thesis, the abnormal measurement percentage is used e.g. the outliers of the sensors historical readings, to calculate this metric. This can also represent the percentage of past successful interactions as described by [106]. Similarly, metrics can be developed that capture conflict information (*C*) by considering the disagreement between the sensors behaviour with other sensors (*O*) and / or background information (*B*). The *O* metric is the percentage of readings of the sensor that conflicts with the measurements of other sensors e.g. sensors that are deployed nearby are of the same type. The metric *B* measures the percentage the sensor produces a measurement that conflicts with background information e.g. a measurement that is practically unlikely for a particular location. The metric *V* (views of experts) represents the average value of the ratings that the users provide based on their knowledge of the trustworthiness of the sensor. In general, Contextual Factor (*X*) metrics can be captured as binary values that are either 0 or 1. They represent an extensible list of factors that affect the trustworthiness of a sensor. For example, the sensor exposure can have an impact on the trustworthiness of a sensor as to whether it is placed indoors or outdoors. Certain contextual factors can also be useful to explain the irregularities of sensor measurements. For

instance, conflicts between sensor measurements due to a nearby factory or conflicts due to sensors not being calibrated.

Table 5.2 provides examples of how individual metrics can be calculated based on available sensor measurements. The trust metrics are calculated using these formulae with the option to be overridden. The views of experts metric, *V* calculates a rating based on a weighted average of a rating given by the users for a sensor measurement instance. In the current implementation, while the calculated metrics e.g. *H, C, V* are real numbers between 0 and 1 the contextual factors (*X*) can have a value of either 0 or 1.

In later sections it is investigated if these metrics are associated with a time frame as to whether they could be reset or updated after a certain period of time. The model is extensible for the incorporation of new metrics. For instance additional metrics can be included by adding the formula for its calculation.

Metric	*Formulation Description*
H	This metric calculates the outliers (formula 5.4) for a set of readings m_i in a time window of size w. The statistics of m_i is used to determine the lower bound L and the upper bound U. Q_1 and Q_3 are the first and third quartile and IQR is the inter-quartile range. If the measurement is less than $L = (Q_1 - 1.5*IQ)$ or greater than $U = (Q_3 + 1.5*IQR)$, then it is an outlier. The ranges are subjected to a tolerance threshold β_H. $$H = \left\{ \sum_{i=1}^{w} f(m_i) \right\} / w \quad (5.4)$$ *Where,* $$f(m_i) = \begin{cases} 0 & if \ ((L - \beta_H) < m_i < (U + \beta_H)) \\ 1 & otherwise \end{cases}$$
O	This metric calculates the conflicts (formula 5.5) of sensor s with neighbouring sensors n. it is formulated by comparing the sensor

Metric	*Formulation Description*
	measurement average $\overline{m_w^s}$ with its neighbour's measurement average $\overline{m_w^n}$ for a time window of size w. A coefficient α_{sn} is a weight based on the spatial distance and is calculated for all neighbouring sensors. This is subjected to a tolerance threshold β_O. The value of β_O can be based on sensor properties described in section 3 (E) plus any additional threshold value. The number of neighbouring sensors is denoted by *k*. $$O = \left\{ \sum_{n=1}^{k} f\left(\overline{m_w^s}, \overline{m_w^n}, \alpha_{sn} \right) \right\} / k \quad (5.5)$$ *Where,* $$f(\overline{m_w^s}, \overline{m_w^n}, \alpha_{sn}) = \begin{cases} 1 \;\; if \left(\dfrac{\left| \overline{m_w^s} - \overline{m_w^n} \right|}{\alpha_{sn}} > \beta_O \right) \\ 0 \qquad\qquad otherwise \end{cases}$$
B	This metric calculates the conflicts (formula 5.6) between a set of sensor measurements m_i and the background information for a time window of size *w*. Each measurement m_i is compared with the minimum (m*in*) and maximum (m*ax*) practical reading at a location (the background information). The ranges are subjected to a tolerance threshold β_B. $$B = \left\{ \sum_{i=1}^{w} f(m_i) \right\} / w \quad (5.6)$$ *Where,* $$f(m_i) = \begin{cases} 0 \;\; if \;\; ((\min - \beta_B) < m_i < (\max + \beta_B)) \\ 1 \qquad\qquad otherwise \end{cases}$$

Table 5.2: The formulations of the Trustworthiness Metrics

5.5.4. Representing Trustworthiness Data in Ontology

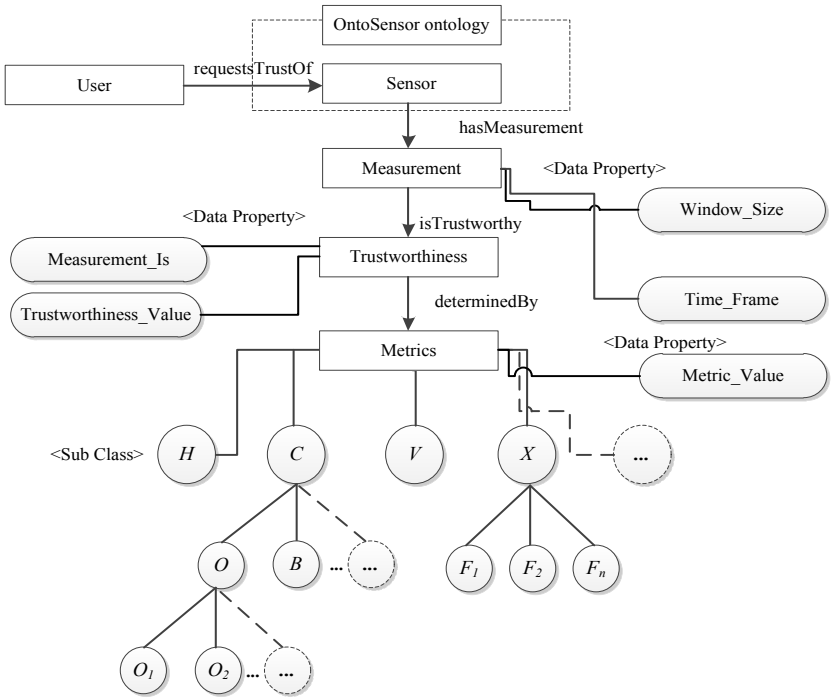


Figure 5.6: The trustworthiness Ontology created by extending OntoSensor

Figure 5.6 illustrates WikiSensing's trustworthiness ontology which is an extension of *OntoSensor*. *OntoSensor* ontology, the top level ontology is depicted using a dotted box. The S*ensor* class of *OntoSensor* is the linking point to the trustworthiness information. The extended trustworthiness classes include *User*, *Measurement*, *Trustworthiness*, *Metrics* and its sub classes (e.g. *H, C,* etc.).

The <*sub class*> annotations on the undirected lines represent the 'sub class' relationships. For example, the classes *H* and *C* are subclasses of the *Metric* class and the classes *O* and *B* are sub classes of *C*. The *object properties* (that links two objects or instances) are shown on the directed arrows and the *data type properties* (links an object with data values) are shown using rounded rectangles.

The trustworthiness model discussed previously is extensible and new metrics can be incorporated. The metrics can be associated with a specific user as well as depend on the time frame or the parameters used for its calculations. Due to

the extensible and diverse nature of the metrics, an ontology is an appropriate method to represent this information. The following code snippet illustrates a subset of classes and properties of the WikiSensing trustworthiness ontology schema.

```
<owl:Class rdf:ID="Trustworthiness"/>
<owl:Class rdf:ID="Measurement"/>
<owl:Class rdf:ID="Metric"/>
<owl:Class rdf:ID="Historical">
    <rdfs:subClassOf rdf:resource="#Metrics"/>
  </owl:Class>
  <owl:Class rdf:ID="ViewsOfExperts">
    <rdfs:subClassOf rdf:resource="#Metrics"/>
  </owl:Class>
  <owl:Class rdf:ID="Contextual">
    <rdfs:subClassOf rdf:resource="#Metrics"/>
  </owl:Class>
<owl:ObjectProperty rdf:ID="determined_by">
    <rdfs:domain rdf:resource="#Trustworthiness"/>
    <rdfs:range rdf:resource="#Metrics"/>
  </owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="is_trustworthy">
    <rdfs:domain rdf:resource="#Measurement"/>
    <rdfs:range rdf:resource="#Trustworthiness"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_measurement">
    <rdfs:range rdf:resource="#Measurement"/>
    <rdfs:domain rdf:resource="#Sensor"/>
</owl:ObjectProperty>
```

A subset of the WikiSensing trustworthiness ontology schema

Restrictions can be applied to enforce certain validations, for example a restriction that the *Trustworthiness* class needs to have at least one Metric to be valid (Figure A.2, of Appendix). This restriction is acceptable as the trustworthiness of a measurement cannot be assessed without at least one metric. The complete ontology is available under trustworthiness ontologies at *wikisensing.org*.

The text in this ontology can be translated as a particular user requesting the trustworthiness of a sensor measurement. The window size and the time frame of the measurement are also listed. The trustworthiness of the sensor measurement is represented using data properties of *'Trustworthiness_Value'* and *'Measurement_Is'*. Moreover the trustworthiness of the measurement instance is determined by metrics and this example shows the metrics of historical information (*H*) and the conflicts with other sensor (*O*). These metric values are represented as

data properties of type double. An *RDF* query language such as *SPARQL* (*SPARQL Protocol and RDF Query Language*) [107] can be used to obtain the triple patterns from this ontology.

5.5.5. The Data Flow

The metric calculation for this example scenario requires data from the sensor databases, *Wiki* pages and the sensor ontology. Figure 5.7 illustrates the data collection for this scenario.

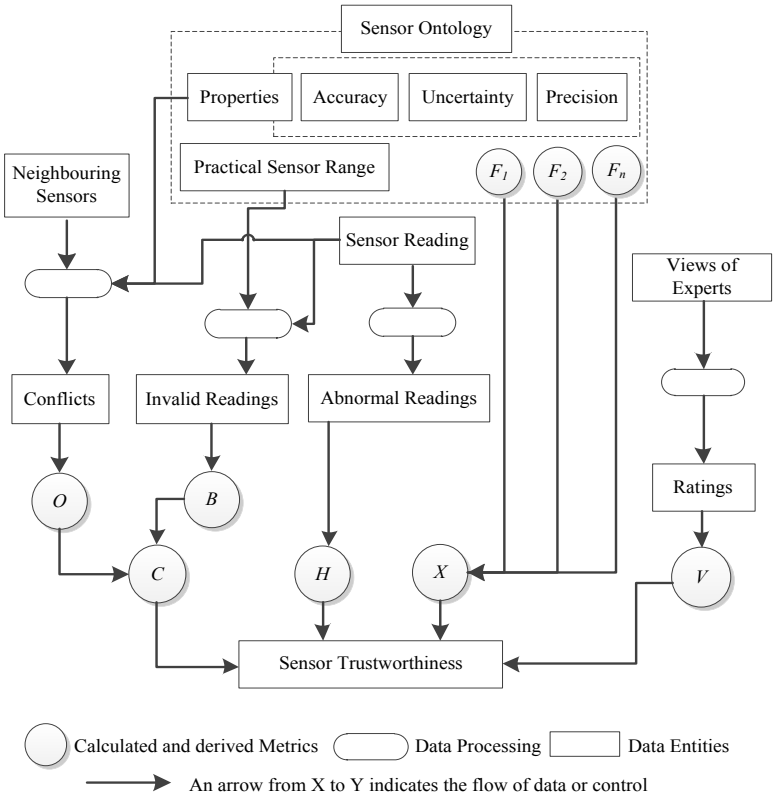


Figure 5.7: The data collection and processing for metrics calculations

To calculate the conflicts with other sensors (*O*) data on the sensor measurements, the neighbouring sensor measurements (neighbour's selected based on radius) and information on the sensor properties from the ontology are obtained. Calculating the background conflicts metric (*B*) requires data on the sensor measurements and information on the practical sensor readings for the particular

location which is obtained from the sensor ontology. The historical metric (*H*) is calculated by obtaining past measurements for that sensor. The views of experts metric (*V*) is calculated by averaging the rating provided on the sensor Wiki pages and the contextual information (*X*) are obtained from the sensor ontology.

5.6. Experimental Evaluation

The objective of the experiments is to evaluate the framework for trustworthiness modelling. The effectiveness of both the Naïve Bayesian and Bayesian Network models is compared in modelling trustworthiness as well as to compare whether the use of continuous metric values (between 0 and 1) or the use of binary variables is more effective. It is also investigated how early the methodology is able to detect untrustworthy sensors once a sensor starts malfunctioning. Finally, the different options for calculating and using the views of expert's information are also investigated.

5.6.1. Experimental Data Sets and Parameters

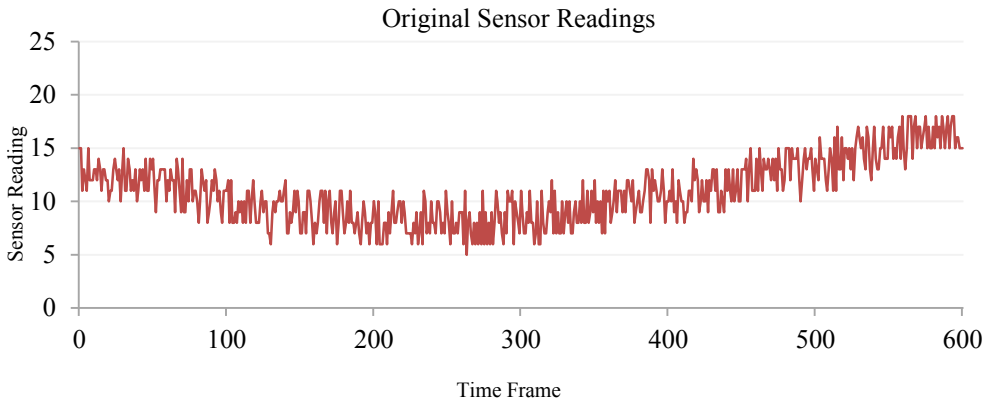
The experimental evaluation is based on the *GUSTO* data set. As the original data set contains only trustworthy measurements from trustworthy sensors to simulate the existence of untrustworthy sensors, a number of specific errors in known sensors are introduced. It is then investigated whether the models and tools would detect these errors or not. Four different scenarios of simulated untrustworthy data are investigated. These scenarios contain sensor readings with large variations, readings that are inactive, readings with temporally-localized abrupt changes and readings with gradual changes.

- *Scenario 1*: This scenario simulates sensors that produce readings with large variance in value (Figure 5.8.b). A value that is two or three times the original measurement is added and subtracted from sensor stream.
- *Scenario 2:* This scenario represents inactive sensors, e.g. sensor readings that

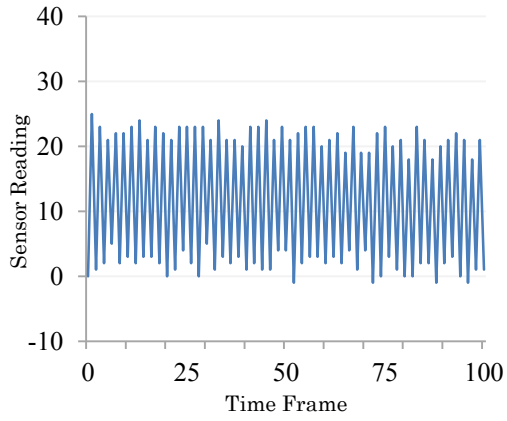
are continuously a constant value (Figure 5.8.c). In some cases the constant value is set to the average of the original data stream.

- *Scenario 3:* In this scenario the data stream values are altered abruptly after a period of time (Figure 5.8.d). A value of two or three times the original measurement is added or subtracted for a portion of the stream.
- *Scenario 4:* In this scenario we gradually change (increase or decrease) the sensor data stream after a certain period of time (sensor 3 in Figure 5.8.e).

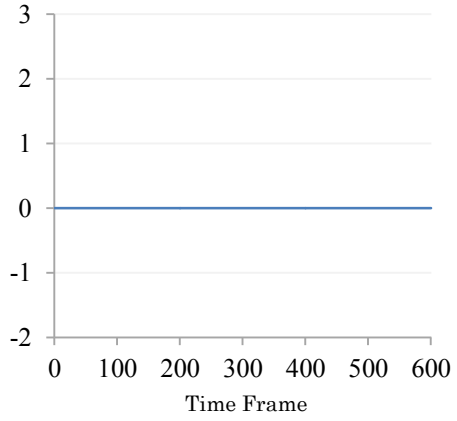
60 sensors of the 560 *GUSTO* sensors are chosen to simulate untrustworthy sensors based on the scenarios; with 15 random sensors per scenario. These sensors are selected from the sensor grid so that they are spatially well spread to avoid clusters.



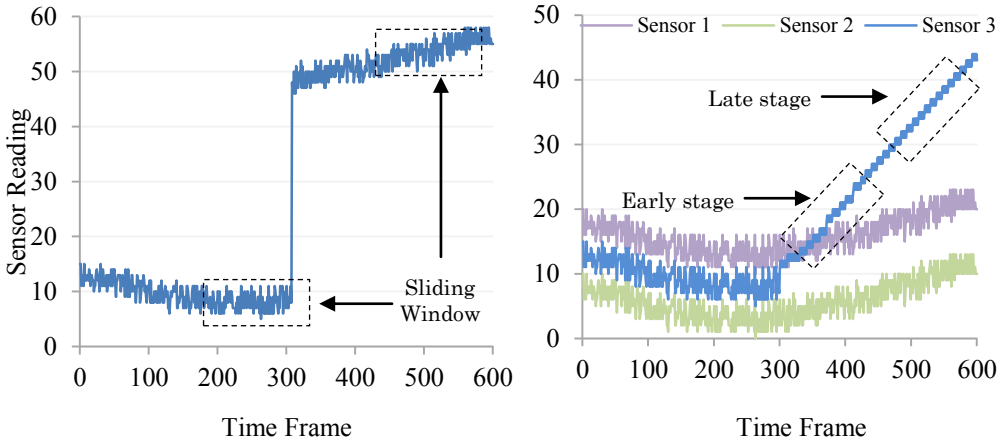
(a)



(b)



(c)



(d) (e)

Figure 5.8: (a) Original sensor readings, Simulations of untrustworthy sensors (b) Large differences in readings (c) Inactive sensor (d) Temporally-localized abrupt change (e) Gradual change

For scenario 1, the change involves a large constant value being added to and subtracted from the entire sensor stream. For scenario 2, the entire sensor stream is set to a constant value. For scenario 3, the change is applied after a specific time with a constant value being added or subtracted to the remainder of the sensor stream. For scenario 4, the change is made after a specific time, however this change is gradual. 1000 random windows each consisting of 100 measurements are taken covering all 560 sensors. These 1000 windows are selected so that there are 700 windows from trustworthy sensors and 300 windows from untrustworthy sensors.

The selection of untrustworthy measurement windows ensured that all important measurements of the change were represented. For example, in the case of a gradual change (Figure 5.8.e) measurement windows were taken from early as well as late stages of the data stream. The selected untrustworthy sensor measurement characterised some typical problems of sensors. For instance, when a sensor broke or got stuck on a particular value or even when the problem was a bit more subtle as in the case where the change is gradual.

5.6.2. Metric Calculation

The metrics are calculated for each selected window that was selected randomly as depicted by the graph in Figure 5.8.d. The *H* metric is calculated using the number of outliers in the data stream and the *O* metric by comparing the sensor measurement with nearby sensors. The minimum and maximum sensor reading that is possible in this area is used to calculate the *B* metric. These values are selected by examining previous pollutant data for this area. The views of experts (*V*) metric are not used for this experiment as that data was insufficient (challenge discussed in chapter 7). The tolerance threshold values of β_H and β_B are set to 0 in order to achieve a higher level of sensitivity and set β_O to the value of the accuracy of the sensors. The calculated metrics (*H, O* and *B*) contains a value from 0 to 1. This value is derived from the percentages from Table 5.3. The contextual data used for this evaluation is based on *Measurement Influencing Factors* (F_I) that may *Exist* (1) or *Non-exist* (0).

Table 5.3 provides a snapshot view of the distribution of metric values for the untrustworthy scenarios as well as trustworthy sensors grouped by different time frames. The time frame column denotes the time instance of the calculation window. Scenarios 1 to 4 represent the non-trustworthy sensors and scenario 5 represents the trustworthy sensors. Moreover time frame 501 is considered as the point of change for scenarios 3 and 4. For example, in scenarios 3 (temporally-localized abrupt changes) the *H* metric is '0' until time frame 500 and then continues to increase and becomes '0' again. The rationale for this change in value is based on the number of outliers or abnormal readings. The *O* metric for scenario 1 (large variances) is consistently 1 as its measurements conflicts with its neighbouring sensors. Moreover the *O* metric for scenario 2 (inactive sensor) can be between 0 and 1 depending on the constant value. However, the *O* metric does not have a value during the early stage of scenario 3 (abrupt change) and 4 (gradual change) as the change is not adequate to trigger a conflict. For scenario 5 (trustworthy sensors), although the *H* and *O* metric may contain values, the *B*

metric is consistently 0. Moreover all metrics in continuous form showed a variance ranged from 0.15 to 0.2.

Metric	*Scenario*	*Time Frame*	*Value*
H	1	0-1000	0
	2	0-1000	0
Abnormal readings in Historical Information	3	0-500	0
	3	501-1000	min 0, max 0.2
	4	0-1000	0
	5	0-1000	min 0, max 0.2
O	1	0-1000	1
	2	0-1000	min 0, max 1
Conflicts with other sensors	3	0-500	0
	3	501-100	min 0, max 1
	4	0-500	0
	4	501-1000	min 0, max 1
	5	0-1000	min 0, max 1
B	1	0-1000	1
Conflicts with Background Information	2	0-1000	min 0, max 1
	3	0-500	0
	3	501-1000	min 0, max 1
	4	0-500	0
	4	501-1000	min 0, max 1
	5	0-1000	0

Table 5.3: Distribution of metric values for sensor categories

5.6.3. Training the Models

The data set is randomly split into training and testing. The training data set contains 500 windows from the trustworthy sensors and 200 windows from untrustworthy sensors; 50 windows per untrustworthy scenario. The test data set contains 200 windows from trustworthy sensors and 100 windows from untrustworthy sensors; 25 windows per scenario.

For the first experiment the calculated metrics and the contextual factors for sensor measurements are used to train the Bayesian models. The contextual information consists of factors that influence the sensor reading (e.g. information on nearby factories or hospitals). The following feature vector (Figure 5.9) illustrates a subset of the training data. The first column (*MI*) is the sensor

measurement instances and is followed by the calculated metrics and the single contextual Factor, F_1, which indicates nearby buildings. Each record is also labelled with actual trustworthiness of the sensor measurement instance. The untrustworthy sensor measurements are labelled as *N* and trustworthy measurements as *Y* in column *T* (Trustworthy).

MI	*H*	*O*	*B*	F_1	*T*
1	0.0	0.0	0.0	0	Y
2	0.0	0.6	0.0	1	Y
4	0.16	0.6	0.0	1	Y
5	0.0	0.3	0.0	0	N
6	0.0	0.6	0.0	0	N
8	0.0	1.0	0.11	0	N
9	0.0	0.1	1.0	0	N
10	0.2	0.0	0.2	0	N

Figure 5.9: A feature vector of a sample set of training data

The Naïve Bayesian model contains all metrics and contextual factors with conditional independence. Two Naïve Bayesian models are developed. The first is based on using the metrics with continuous values. The second is based on converting the metric values into binary. For the metrics *O* and *B* values greater than 0.5 (50%) is set as 1 and for the *H* metric values greater than 0.1 (10%) is set as 1. This disparity exists due to the calculation of the *H* metric, as when the outlier count exceeds 20% it is no longer considered an outlier. Measurements with unique metric combinations (in binary representations) are grouped into the same sensor measurement instance. The Naïve Bayesian model with continuous data is also used in the evaluation with the real values of metric used for training. A classifier software by *Microsoft Research* [108] is used for the Naïve Bayesian.

Figure 5.10 illustrates the specific Bayesian network used for this evaluation which is based on the model previously designed by an expert (Figure 5.2). The data set contains only information on the F_1 contextual factor, used to signify the conditional dependency between impacting factors and other metrics on the trustworthiness, but not other factors, which leads to the simplified network.

Also note that although the *V* metric is not used in this occasion, but is later used in the experiments discussed in chapter 7. The *AgenaRisk* tool [109] is used for the Bayesian network modelling to represent the network and to model the trustworthiness of the sensor measurements based on the input data.

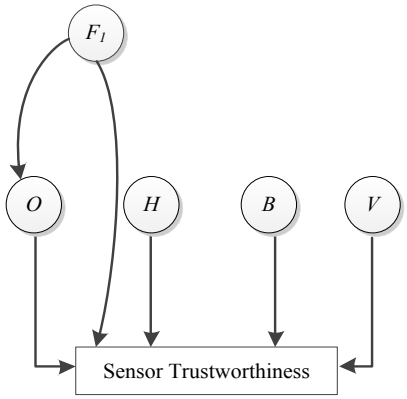


Figure 5.10: The specific Bayesian Network trust representation used for evaluation

The following example equation returns the probability of sensor measurement *m* being trustworthy provided the metrics, *O* is 1, *H* is 0, *B* is 0, *V* is 1 and F_1 is 1 using the classifier in Figure 5.10.

$$P(m = \text{Trustworthy} \mid O = 1, H = 0, B = 0, V = 1, F_1 = 1) = \frac{P(m=\text{Trustworthy}, O=1, H=0, B=0, V=1, F1=T)}{P(O=1)P(H=0)P(B=0)P(V=1)P(F1=1)}$$

Figure 5.11 shows the confusion matrix for all Bayesian models with training data. This is illustrated for the sole purpose of comparing with the results obtained using the test data.

	Predicted					
	Naïve Bayesian (Binary)		Naïve Bayesian (Continuous)		Bayesian Network (Binary)	
Actual	T	N	T	N	T	N
T	TP 500	FN 0	TP 500	FN 0	TP 500	FN 0
N	FP 45	TN 155	FP 28	TN 172	FP 24	TN 176

Figure 5.11: The confusion matrix for Naïve Bayesian (binary), Naïve Bayesian (continuous) and Bayesian network (binary) with training data

5.6.4. Applying the Models on Test Data

The Naïve Bayesian (binary and continuous) and the Bayesian Network (binary) models are compared on test data to evaluate their performance as well as to check for early detection of trustworthiness.

5.6.4.1. Comparing Bayesian Model Strategies

The confusion matrix Figure 5.12 summarises the results obtained by applying the Naïve Bayesian and the Bayesian network models with binary data as well as the Naïve Bayesian model with continuous data. Moreover Figure 5.13 illustrates the entire outcome based on percentages of the results. The accuracy of the tools for the test data is 100% for true positives and 65% for true negatives in the Naïve Bayesian Model. The accuracy of the true negatives has improved to 85% when using the continuous data. The accuracy when using the Bayesian Network is 100% for true positives and 87% for true negatives.

	Predicted					
	Naïve Bayesian (Binary)		Naïve Bayesian (Continuous)		Bayesian Network (Binary)	
Actual	T	N	T	N	T	N
T	TP 200	FN 0	TP 200	FN 0	TP 200	FN 0
N	FP 35	TN 65	FP 15	TN 85	FP 13	TN 87

Figure 5.12: The confusion matrix for the trustworthiness using test data for Bayesian model strategies

Figure 5.14 shows the number of false positives for the Bayesian model strategies based on the untrustworthy scenarios. For scenario 1 (large variance), all three strategies detected 100% of the unworthy sensors correctly. The Naïve Bayesian model with binary data incorrectly identified certain sensor

measurements for scenario 2 (inactive sensor) and scenario 3 (abrupt change). This is when the change is made within the limits of a possible trustworthy measurement and the position of the calculation window does not pick any abnormalities. However the Naïve Bayesian model with continuous data and the Bayesian Network model with binary data achieved an improved rate for these scenarios.

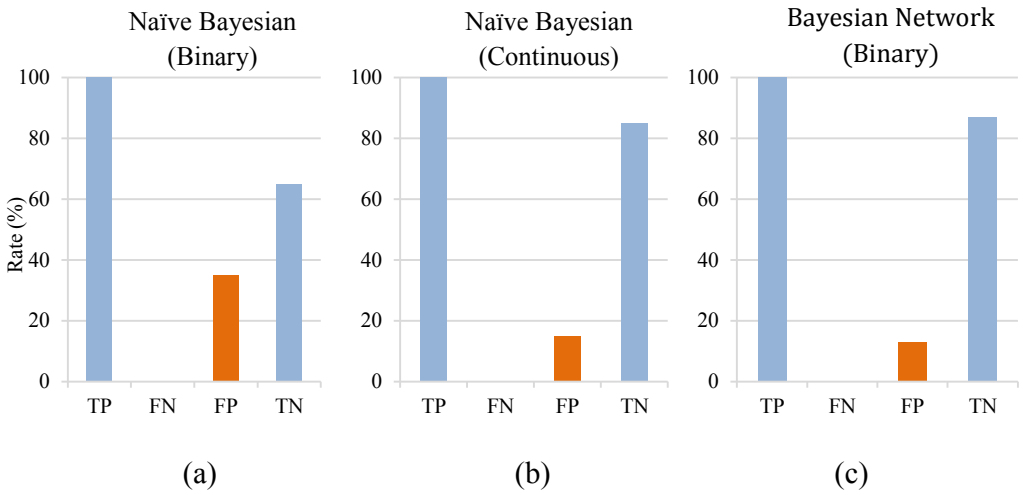
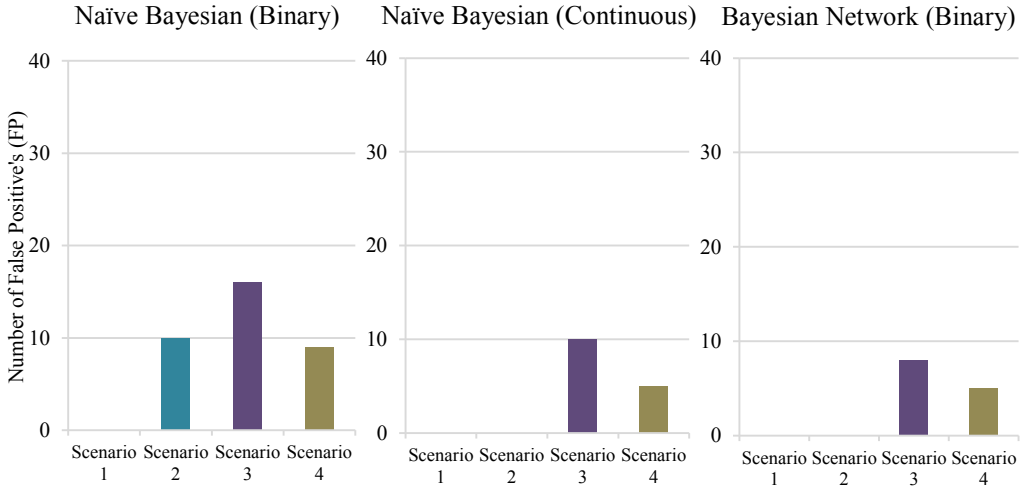


Figure 5.13: Summary of results (percentages) for test data

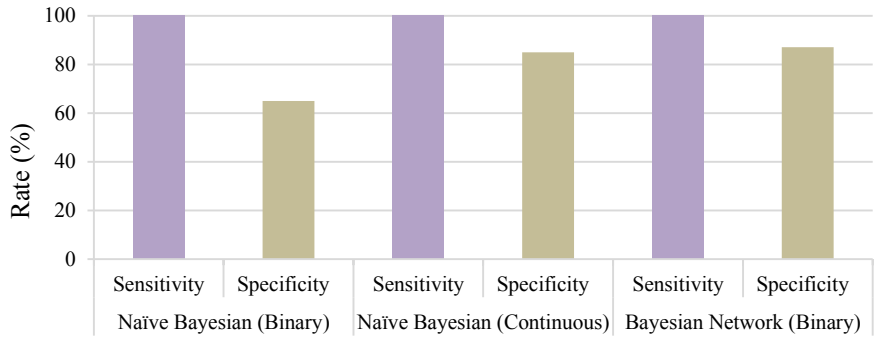
Occurrence of False Positives (FP) based on non-trustworthy scenarios



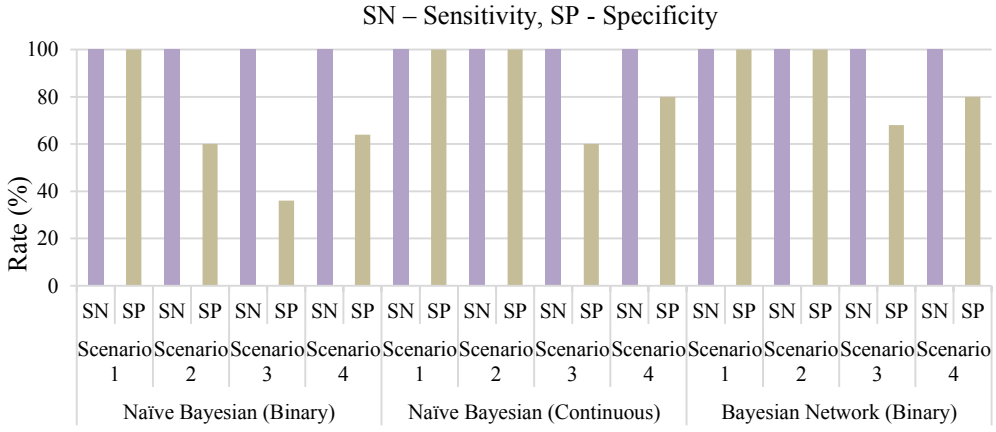
Scenario 1: Large variances, Scenario 2: Inactive sensor,
Scenario 3: Abrupt change, Scenario 4: Gradual change

Figure 5.14: Distribution of false positives for untrustworthy scenarios

The Naïve Bayesian model with continuous data has an improved rate as it considers the actual metric values which are otherwise lost when converted to the binary form. The Bayesian Network model with binary data obtains better results compared to the Naïve Bayesian model as it takes into account the conditional dependencies between the metrics. For example this strategy is able to capture the conditional dependencies between the contextual factors and other metrics. For scenario 4 (gradual change), all models do not correctly identify the early stage as all calculated metric values are 0.



(a)



(b)

Figure 5.15: (a) The Sensitivity and Specificity rates for Bayesian models (b) Distribution of sensitivity and specificity rates for untrustworthy scenarios

$$Sensitivity = \left(\frac{number\ of\ True\ positives}{number\ of\ True\ positives + number\ of\ False\ nagatives}\right) \quad (5.7)$$

$$Specificity = \left(\frac{number\ of\ True\ negatives}{number\ of\ True\ negatives + number\ of\ False\ positives}\right) \quad (5.8)$$

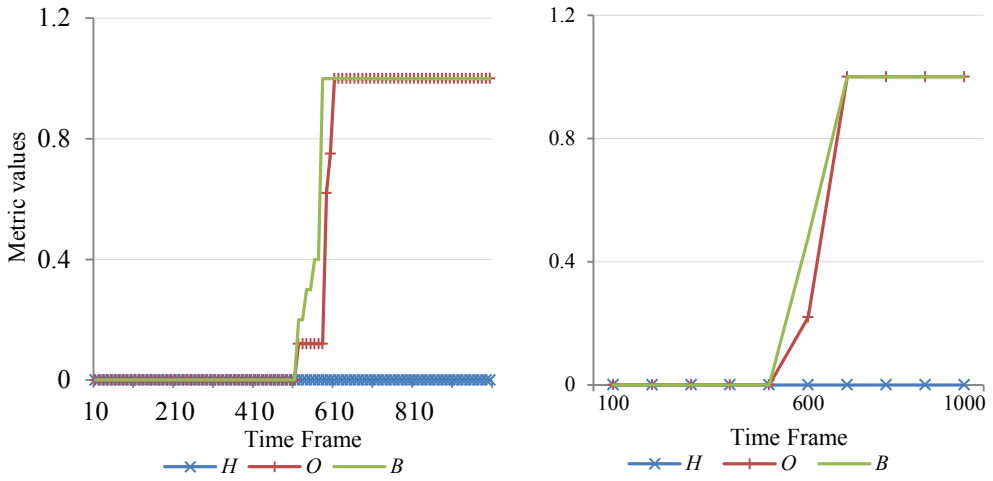
The sensitivity (formula 5.7) and specificity (formula 5.8) values for the Bayesian models are illustrated in Figure 5.15. All models display a high sensitivity rate and if the sensor measurement or measurement window is determined as untrustworthy it can be certain that it will not be accepted by the user to be correct. A high specificity is also demonstrated in all three models hence if the sensor measurement instance is determined as trustworthy it can be certain that this measurement will be accepted by the user to be correct.

5.6.4.2. Evaluating Early Detections

For the second experiment, the Naïve Bayesian models are trained with binary and continuous data using a measurement window of 10 measurements as opposed to 100 measurements used in the first experiment. The aim is to obtain the metric values with a lower granularity to identify the point when the sensor is detected as untrustworthy. The data used for this experiment is the same data set that was used in the previous experiment. Moreover the untrustworthy scenario 4 (gradual change) is tested by calculating the metrics for a smaller window of 10 as well as 100 measurements for the entire sensor. The rationale of using scenario 4 is that it is the only scenario that exhibits a continuous gradual change to sensor measurements.

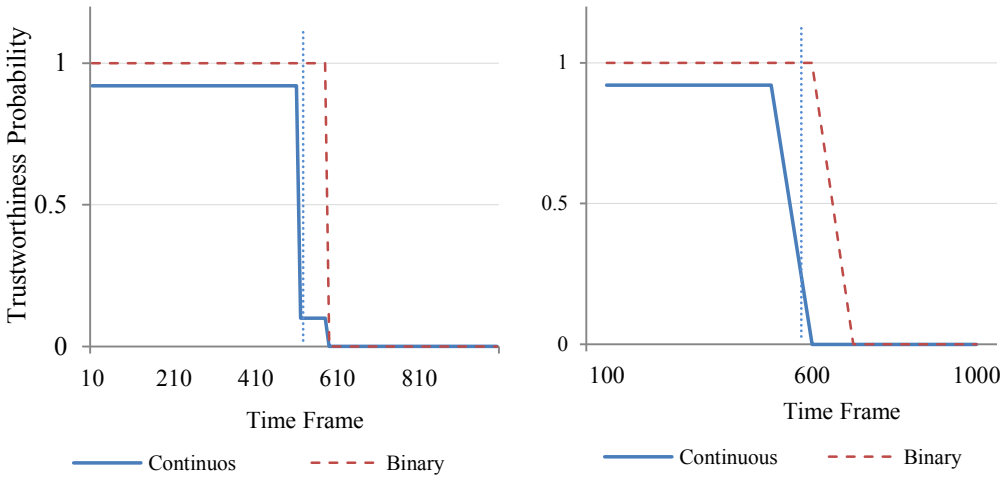
The metrics that are used for this experiment change with respect to time. Figure 5.16 a and b shows the comparison of values for the metrics of *H, O* and *B* for the untrustworthy sensor simulated by scenario 4 (Figure 5.8.d) for a window sizes 10 and 100. Figure 5.17 a and b shows the trustworthiness probabilities of this sensor calculated based on the same metrics for the Naïve Bayesian model using both as binary and continuous representations of the metrics. The results

demonstrate the advantage of using continuous values and smaller window size for the metrics as untrustworthy sensor measurements are detected much earlier.



(a) (b)

Figure 5.16: The *H, O* and *B* metric values for one sensor in scenario 4 with calculation window of (a) 10 measurements (b) 100 measurements



(a) (b)

Figure 5.17: The trustworthiness probabilities by applying Naïve Bayesian model with continuous and binary values for untrustworthy scenario 4 (a) window size 10 (b) window size 100

The vertical dotted line at time frame (500) indicates when the sensor readings started producing anomalous values. When using continuous values both the *O* and *B* start increasing as conflicts with other sensors and also background information is detected by the model. This can be detected only at time frame 610 when the binary metric is used. Since the change is gradual the *H* metric does not change. Throughout the period the *H* metric is not affected as the change is gradual.

5.6.5. Result Discussion

Overall, the results of all the experiments using the sensor metrics are encouraging. However, high number of false positives resulted in scenario 3 and 4 for all models, with false positives resulting in scenario 2 for Naïve Bayesian model with binary data. For scenario 2 it's the case when the continuous constant value is within the range of the background data as well as when it does not conflict with its neighbouring sensors. In scenario 3 and 4 the false positives are when the measurement windows are taken at an early stage.

A solution for the problems associated with scenario 2 is to identify when a sensor reading stays stuck for a long time, and for scenario 3 to detect when a measurement suddenly drops more than a certain percentage, and a solution for scenario 4 is to recognize the upward or downward trends of sensor measurements. The increase in the sensitivity of the metric values when using continuous data as opposed to binary is also an option. A solution to increase the accuracy by reducing the false positives would be to increase the tolerance parameter (β values) when calculating the metrics. However this can end up compromising other sensor streams that do not require such tolerance making it a trade-off. Another option is to add more contextual data as well as to change the dependencies among the metrics especially in the Bayesian Network strategy.

The Bayesian Network model obtained better results compared to the other Bayesian models as it is clear that conditional dependencies can exist between the

metrics. Moreover the dependencies in the Bayesian Network are also useful in situations when certain metric values are unavailable or not known. For example, the arrow from F_1 to O (Figure 5.2) determines that conflicts between measurements (O) are influenced by impacting factors (F_1). Hence we can infer that if there is an impacting factor, there is a possibility that the measurements can conflict with measurements of other nearby sensors.

The results in Figure 5.17 for the second experiment demonstrate the advantage of using continuous values as opposed to binary values with the Naïve Bayesian model. This is due to the increase of the sensitivity of the metrics which is lost when the metrics are converted to binary values in the other strategy. Moreover it also shows the advantage of using a smaller metric calculation window as the untrustworthy sensor measurements are detected much earlier.

5.7. Related Work

The definition of trust formulated in this research relates to the definitions in [110] that is based on previous evidence as well as the definition by [63] that is based on previous actions. Conversely the trust definition of this thesis is based on past and current metrics and contextual data that represent the behaviours of the sensors. The framework proposed by [54] uses reputation metrics to assess the trustworthiness of sensors. However this framework is applied for sensor networks and the metrics only consider the discrepancies of sensor measurements. In contrast the WikiSensing trustworthiness framework can be applied to any sensor and the metrics are calculated on previous, current sensor measurements, contextual data, views of experts and conflicts with neighbouring sensors or background data. Moreover this model is extensible so that new metrics can be incorporated when needed.

5.8. Conclusion

This chapter investigated the challenges of managing trustworthiness in WikiSensing and presented a framework and methodology based on a generic probabilistic definition of trust. It described how to capture and calculate metrics for different types of available evidence. The approach is extensible allowing incorporating metrics based on other probabilistic models if needed.

The experiments demonstrate and verify the use of the framework and models and also compared different representational Bayesian models. The key advantage of employing Bayesian modelling approach is that it provides a natural way of combining prior information with data to predict future outcomes (posterior) in a probabilistic manner. The Bayesian Network model used in the experiments provided more accurate results when compared with the Naïve Bayesian model which is much simpler. In addition the advantage of the Bayesian Network model is that it captures conditional dependencies and enables prediction of certain metrics when the values were not known. It also allows a more hierarchical definition of such relationships. There is scope to continue exploring the advantages of such networks in the future. The Naïve Bayesian model with continuous values provided better results than when using binary values. This is due to the loss of information when converted to binary values.

It was also noticed that the use of continuous values for metrics improved early detection of untrustworthy sensors due to the increase of sensitivity of metric values. This was a trade-off as certain situations did not require such sensitivity. Moreover smaller calculation windows also resulted in early detection of untrustworthy measurements.

The anomalies that were introduced in the test data are not exhaustive. When using binary values for the metrics, the number of possible configurations is clearly limited in contrast to when using continuous values. However, the advantage of having a fixed set of configurations is that they are more robust and effectively become a base set of rules in deciding whether the sensor measurement

is trustworthy or not. It is planned to further explore the use of continuous values and to control sensitivity to achieve a higher accuracy in determining outcomes.

Moreover it must be noted that the data used for trust assessment in this work is archived data that were obtained from sensor data streams. Hence it will also be interesting to investigate the assessment of trust with real-time data. One approach will be to recalculate the metrics and the trust ratings as new sensor measurements arrive. Another approach will be to only recalculate the values on certain time frames or measurement count intervals to avoid the overhead of calculating metrics and trust ratings for each new sensor measurement.

Furthermore the *Dempster-Shafer* theory of evidence [111] which is based on Subjective logic [112] is an approach to combine evidence from different sources and determine a degree of belief. The degree of belief is represented as a belief function which contrasts to Bayesian theory that deals with probability distributions. This approach can be easily incorporated into the current framework to determine belief (trustworthiness) by composing evidence (metrics).

6. Integrating Expert Knowledge

WikiSensing has been and continues to be used in a wide range of applications that benefit from the flexible data management functionalities as well as its trustworthiness assessment features. The system continues to support the data management of live sensor data, the assessment of trustworthiness of sensors as well as facilitate other data analysis frameworks.

The work by *Cano et al.* [113] demonstrates that integrating expert knowledge can be useful to overcome problems such as learning of Bayesian Networks from data when the data are scarce as well as when problem domains contain a high number of random variables. Moreover the notion of expert knowledge is also important to WikiSensing on the basis that it encourages collaboration with the aim of obtaining user feedback and annotations from experts on sensor data. This chapter presents several case studies that demonstrate the usage of WikiSensing with the intension of demonstrating how expert knowledge can be integrated.

The first case study that is discussed is based on WikiSensing supporting the 2013 *UPLondon Hackathon* and *Crackathon* events by providing data management and trustworthiness assessment functionalities. The system also provided access to a set of live data stores such as *TFL* and *MetOffice* with querying functionalities through its *API's.* The second case study describes how WikiSensing is managing route data for the visually handicapped that are collected by researchers at the Bio-Engineering department at Imperial College, London since early 2013. This real-time data is sent to the system by various sensor devices (e.g. accelerometers, gyroscope, etc.) with heterogeneous data formats.

6.1. The UPLondon Hackathon and Crackathon

WikiSensing was one of the main data management platforms and data stores that supported the *Hackathon* and *Crackathon* events at the *Urban Prototyping London* (*UPLondon.org*) festival in April 2013. For the *Hackathon*, WikiSensing hosted meteorological data of cities around Britain, transport data on traffic disruptions and tube departure boards and device-level electricity usage data. To ensure reliability during this 3-day event the system was stress-tested using 1000 concurrent users and deployed a back-up cloud infrastructure on *Windows Azure* [114]. The objective for the participants was to create cutting-edge technology solutions that result in real-world change, based on the environment, local economy or local community.

The trustworthiness of sensor data was explored during the *Crackathon* events by testing WikiSensing's trustworthiness *API* with external users. For the *Crackathon*, contestants were given air pollution data of an area in East London, which had been selectively altered in different ways, to simulate potential attacks. The task was to assign a trustworthiness score to measurements of different sensors at different time frames, with the aid of WikiSensing's trustworthiness API which offered history-based abnormal reading detection and neighbour-based conflict detection.

6.1.1. The Hackathon event

WikiSensing provided data management services for the *UPLondon Hackathon* event (*sustainablesocietynetwork.net/th_event/hackathon*). The participants were given access to a number of comprehensive data sources that were collected at real-time by WikiSensing. These include *Meteorological Office*, temperature and wind speed data (two weeks), *Transport for London,* tube boards and traffic disruptions data (two weeks), and household device-level electricity usage data (three years) that was monitored by a group of researchers at Intel. WikiSensing provided a set

of services (*wikisensing.org*) for querying this data as well as services for participants to create their own sensor data sources.

The provided data sources proved to be a valuable source of information for potential new applications. There was keen interest in the amount of detail stored on London transport data. The ability to manage heterogeneous record types in WikiSensing was a key factor in the flexibility of application development during this event.

6.1.2. The Crackathon event

The main goal of the sensor data trustworthiness assessment task during the *Crackathon* event was to understand how users rate sensor measurement under certain conditions. During this event the participants were given two data sets of sensor measurements where some contained alterations. The data provided were managed by WikiSensing and the users were given access to its data management services to query this data as well as trustworthiness services (Figure A.3 and Figure A.4 of Appendix) to generate trust metrics. The participants' task was to detect the changes in the sensor data and rate the trustworthiness of a set of sensor measurements at specific time frames. Figure **6.1** illustrates a scenario where the trustworthiness of sensor data can be jeopardised. The actors are denoted using dashed boxes. The custodian is WikiSensing that manages the sensor data. The owner submits sensor data to the system and the user access these sensor data streams. Furthermore the attacker falsifies the sensor data streams stored in WikiSensing.

Two separate sets of pollution data were used for this activity. This data was obtained using *GUSTO* sensors that monitored the pollution levels at a busy location in East London. The two data sets were distinguished by sampling the measurement to different frequencies. Both these data sets contained four different types of pollutants (*NO, NO_2, SO_2* and *ozone*). Figure 5.5 (a) illustrates the sensor deployment. Contextual information were also provided (Figure 5.5 (b)) on the

location of the pollution sensors, as well as historical measurements. Tools were provided to allow users to review the data sets as well as to assess trustworthiness of sensors by discovering anomalies and conflicts in the data (e.g. historical data, contextual data, and conflict with other sensors).

The attacker was simulated by introducing alterations to the sensor data streams. In each case one of the data sets would have been changed (e.g. tampered by changing a subset of measurements). Some of these changes were obvious while the others were subtle. Moreover the changes were either an abrupt or a gradual change in the data, with the change being applied to a single or multiple (possibly correlated) data streams. Importantly the participants were not aware which data was changed.

The objective for the user (participant) was to first detect when the attack occurred and to identify which sensors were attacked. The participants were also requested to report the "correct" pollution values at specific locations and justify the value they chose if the sensors in the same location report different values. The following are the strategies used for sensor measurement alterations.

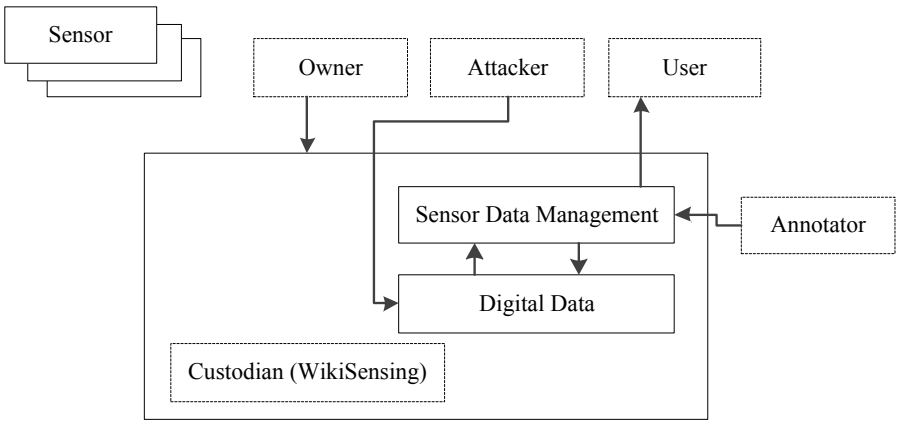


Figure 6.1: The potential actors involved in sensor data management

Case 1: Inactive or faulty sensor

The sensor measurements were set to a constant value throughout the entire stream to simulate an inactive sensor (Figure 5.8.c). Moreover a constant value was added or reduced from the measurement to replicate faulty sensors (Figure 5.8.b).

Case 2: Temporally-localized abrupt change in a single-sensor data stream

The data stream values were abruptly changed after a period of time and is similar to the data stream depicted in Figure 5.8.d. This change was only being applied to one sensor and the changes to the data stream were more or less apparent. Consider, for example, a sudden increase or decrease in the value range of the data stream, or the introduction of a sharp peak.

Case 3: Gradual change in a single-sensor data stream

The data stream values were gradually changed after a certain period of time. This was applied to a single sensor and the change was not easily identifiable as the previous scenario. This alteration is similar to the data stream depicted in Figure 5.8.e.

Case 4: Coordinated change in multiple (correlated) data streams

Gradual changes were applied to multiple sensors and were coordinated across these data streams. This scenario models the need to compare measurements with nearby neighbouring sensors.

6.2. Managing Routes for the Visually Handicapped

The *Royal National Institute of Blind People* indicates improved mobility of the blind and partially sighted as one of its main aims. Hence researchers frequently work on methods that would enable members of the *BPS* (blind and partially-sighted) community to travel safely and independently in indoor and outdoor environments.

Researchers at the Bio-engineering department of Imperial College, London has implemented a system that is capable of capturing navigational paths relying on built-in sensors of mobile devices along with the measurement of *Wi-Fi* signals in a building. Moreover, contextual data such as landmarks and obstacles

(e.g. staircases, traffic lights, revolving doors) are also identified. The mobile application software used for this purpose is implemented on *Android-based* devices, namely the *Samsung Galaxy SIII* and *Nexus 4* phones as well as the *Nexus 7* tablet. This application serves as a platform for data collection. The gathered data is stored and is then queried from a remote machine via the WikiSensing *REST* API. Communication between client and the WikiSensing server is implemented via HTTP, and files transferred using XML.

An initial request is made to the WikiSensing server to register the user and the sensors implemented by the application. The details obtained are stored in a new directory on the mobile devices internal memory, and used on every successive run (a unique user ID is given to each device). The URL created with the user and sensor IDs are used to send requests to WikiSensing. The following pseudo-code explains the registration of a user and a sensor in WikiSensing.

```
if userFileDirectory exists then
    userID = parse (userFile);
    sensorID = parse (sensorFile);
else
    send request;   // to WikiSensing
    get response;   // to WikiSensing
    create Directory (userFileDirectory);
    create File (userFile) = get response (userResponse);
    create Directory (sensorFileDirectory);
    create File (sensorFile) = get response (sensorResponse);
    userID = parse (userFile);
    sensorID = parse (sensorFile);
end
URL = wikiSensingURL + userID +sensorID;
```

The *userFile* and *sensorFile* contains a unique user service key and sensor id's. As soon as data starts recording, a timestamp is requested from a *Calendar* class, to create a unique experiment ID under which all the collected data are saved on WikiSensing. As sensor events and Wi-Fi state changes occur, the collected data

is written directly into XML files in the appropriate format. A counter is set so that every 100 sensor events, the three sensor XML files (accelerometer, magnetometer and gyroscope) are posted to WikiSensing (Figure 6.2) and is reset. However since a *Wi-Fi* sample contains much more data than a sensor event, the *Wi-Fi* XML is posted and reset every 10 samples. This ensures that the XML files do not become too big, causing the application to crash. The *Wi-Fi* data is acquired through a Broadcast Receiver, which is registered to receive intents containing information regarding the *Wi-Fi* state of available Access Points.

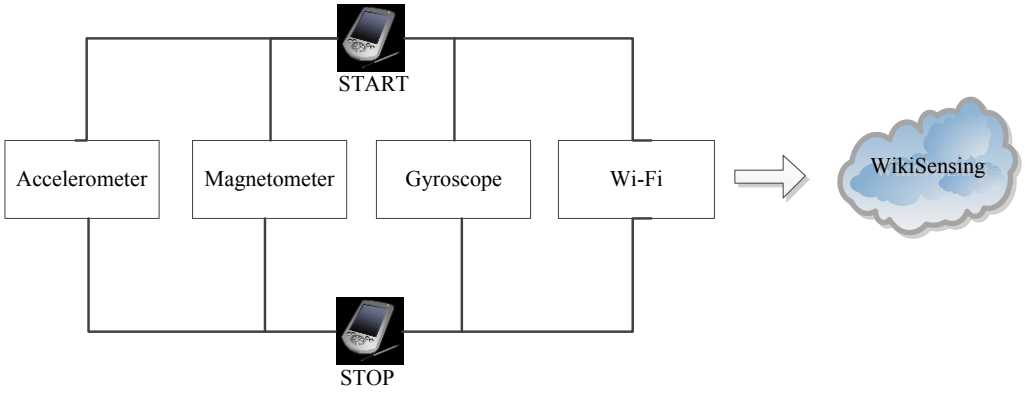


Figure 6.2: Schematic of the application processes

The data on WikiSensing platform is arranged in a hierarchical way shown in Figure 6.3 and every element corresponds to a node of the tree. *parseInSens* is a function that is implemented in *Matlab* to query data from sensors, and arrange the data in cells conveniently for further processing.

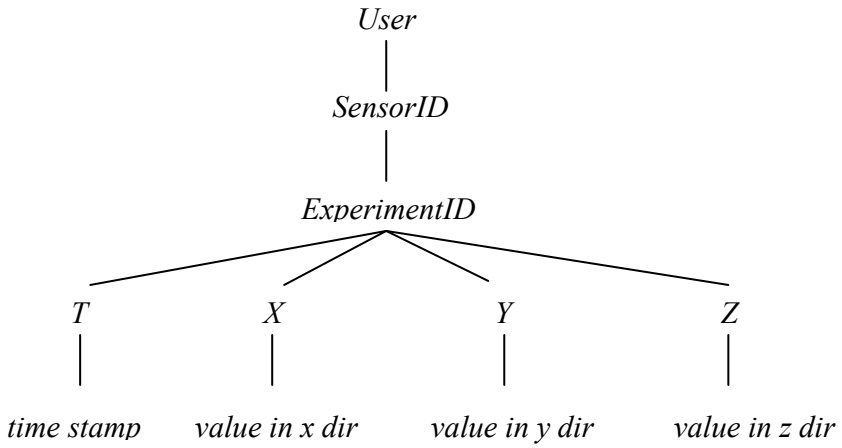


Figure 6.3: Query hierarchy supported by WikiSensing

The First step in mining data is based on constructing a URL. This is accomplished by a *constructURL* function which takes the *userID*, *experimentID* and sensor of interest and outputs a URL which can be then fed as an input to the parsing functions. *parseInSens* accesses each leaf of the tree and puts it into one of 4 vectors, depending on whether it is a time stamp, or a sensor reading belonging to the x, y or z channel. The output is a cell with two elements: a vector with N time stamps and 3-by-N matrix with the corresponding sensor readings in three directions.

According to the users feedback the main advantage of using the WikiSensing platform is that it supports an extensible list of data per record. In addition the ability to store data in a hierarchical structure was beneficial, as the data retrieval from a remote machine can be done by querying at different levels of the hierarchy, depending on what is needed. For example, a user who wants to analyse all the data for one *sensor ID* over all experiments may query by that *sensor ID*; if instead the user wanted to study the behaviour of all *sensor ID's* in a single experiment, they may query by *experiment ID*. The user expects WikiSensing to become more functional in future, allowing for users to store and process algorithms on the server side.

6.3. New Challenges

These case studies demonstrated the versatility of WikiSensing applied to a set of diverse requirements. WikiSensing is used for sensor data management as well as trustworthiness assessment. The *Crackathon* event at the *UPLondon* demonstrated that it is difficult to obtain user rating for a large set of sensor data. Moreover although WikiSensing successfully managed route data it is however a challenge to manage the trustworthiness of such data due to its complexities. The next two chapters investigate these challenges with chapter 7 focusing on estimating user rating for larger sets of sensors and chapter 8 focusing on assessing the trustworthiness of route data.

7. The Views of Expert metric in the Trustworthiness Model

User ratings or user feedback are an integral part of many online systems. This information is commonly used to validate data, to identify trends, as well as applied in predicting similar outcomes. Two popular strategies of obtaining and understanding patterns of user feedback are known as collaborative rating and collaborative filtering.

Collaborative rating system obtains feedback from its users based on the opinion of correctness. For instance, online systems such as *StackOverFlow* and *BioStack* have proved this to be a powerful tool that enables the sharing of opinion as well as the validation of information. On the other hand recommender systems (also known as collaborative filtering) such as *amazon.com* and *netflix.com* encourage user collaboration in order to rate products or media as well as record past interaction with the system. Moreover collaborative filtering is the process of filtering for information or patterns using reviews and observations to predict the viewpoints of users and make future recommendations [115]. Recommendations can be in the nature of preferences of, a watched movie, a read magazine, book or even on a driving route. In a typical recommender system users provide recommendations as inputs, which the system then aggregates and presents information to appropriate recipients [47].

This research considers the views of experts as a rating provided by the users on their acceptability of the sensor data as correct. This is in line with the definition stipulated in section 5.2. The process followed to calculate the views of experts metric in WikiSensing uses principles of both a collaborative rating system

and a recommender system. It first obtains the feedback from users with the intension of understanding their opinion and secondly it estimates the rating based on previously gained feedback when the data are inadequate. The aim of using this rating is to include it in the trustworthiness model to help determine if the sensor or sensor measurement can be trusted.

7.1. The Requirements (challenges)

The view of experts metric (*V*) is a trustworthiness score on the sensor measurement based on the ratings provided by users. Users are able to rate the measurements based on their own experience and knowledge of the sensor or by using other metric values calculated for that measurement instance (e.g. *H, O, B,* etc.). A rating between 1 and 10 can be provided by users. The *V* metric is calculated using the weighted average of these ratings and is labelled as 'Positive' (Trustworthy) or 'Negative (Not-trustworthy). The *V* metric is then incorporated into the trustworthiness model along with the other metrics (e.g. *H, O, B* etc.) to assess the trustworthiness of the sensor measurement.

The main challenge of calculating the views of expert metric (*V*) is that it is not practical to collect ratings for all sensor measurement instances. Hence there can be a lot of unavailable data (non-rated measurements). Three strategies to address this problem based on the way the *V* metric can be calculated for a large set of data are discussed. The first strategy extrapolates the *V* metric using available ratings to reflect on the entire sensor. The second strategy estimates the *V* metric using already rated measurements by following similarities based on other metrics. The third strategy by default sets the *V* metric as '*Unknown'* for every sensor measurement until it is rated by a user.

7.2. Strategies for Modelling Views of Experts

Users can express their views on the form of rating, by voting or even by commenting. When deciding on the trustworthiness of sensors, users can provide ratings based on their knowledge of the sensor, its deployed environment as well as observation on the sensor measurements. This can be an important factor that would help determine the actual trustworthiness of a sensor or a sensor measurement.

The work by [116] contains a useful definition on predicting rating that is based on the heuristic that people who agreed in the past will probably agree again. This is an important aspect in sensor ratings as this thesis considers the profiles of other metrics that were present when the user rated a particular measurement.

Recommendation or collaborative filtering can be applied to rated sensors or measurements in order to understand the preferences of the users e.g. whether users might accept or reject the sensors or the sensor measurements. *Breese et al.* [48] define the task of collaborative filtering to predict the utility of items for a particular user (the active user) based on a database of user votes from a sample or population of other users (the user database). The approach used in this research estimates and extrapolates the views of expert ratings for a sensor measurement using previous ratings obtained by other users. *Breese et al.* also explain the concepts of explicit and implicit voting. The former refers to the user expressing preferences on a title based on a discrete numerical scale and the latter relates to a ratio of information access patterns carried out by the user. Explicit voting is used as the mode of feedback in WikiSensing as it concentrates on the user's preference on the data and not their access patterns.

$$\bar{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j} \qquad (7.1)$$

$$p_{a,j} = \bar{v}_a + k \sum_{i=1}^{n} w(a,i)(v_{i,j} - \bar{v}_i) \qquad (7.2)$$

Memory based algorithms discussed by *Breese et al.* predict (formula 7.2) the preference of user *a* for item *j* which is $p_{a,j}$ based on user votes $v_{i,j}$ (by user *i* on item *j*), the mean vote of user *i* for a set of items *I* (formula 7.1) and a set of weights ($w(a,i)$). This is applied to the total number of users in the collaborative filtering database (*n)* and subjected to a normalising factor *k*. The weighting ($w(a,i)$) used in this algorithm is based on either correlation or vector similarity.

The views of experts metric for sensor data are based on ratings of a specific sensor for a measurement instance which compares to an item in the memory mapped algorithm. Moreover the distinction of the user is not significant as the focus of this thesis is to estimate a user's trust rating for a sensor measurement. One of the other methodologies for estimating the views of experts used here is based on vector similarities of metric patterns. The overall aim of this method is to estimate the trustworthy ratings of sensors or sensor measurements based on previous user ratings which are similar to the votes ($v_{i,j}$) used in the memory based algorithm.

To summarise the way WikiSensing determine the views of expert metric is similar to the concept of collaborative filtering as it deals with missing or unavailable data. However the intension here is to estimate the ratings of other unrated measurements and not to make any recommendations to users. The options that are discussed for estimating the views of experts are based on extrapolating data, modelling existing patterns and adding default ratings.

7.2.1. Extrapolate Views of Expert Metric with Sensor

When the data to calculate the views of expert's metric is unavailable, already obtained ratings are used to estimate it by extrapolation. Two options are explained on the nature of the extrapolation. While the first option extrapolates values to the entire sensor the second option extrapolates values from the time frame of the available user rating.

Option A: The first option extrapolates the rating of the sensor in its entirety instead of a measurement instance. For example, when a measurement of a sensor is rated by users to be '*Negative'* (Not-Trustworthy) all its measurement instances will be rated as '*Negative'*. By default (before any rating is set by user) a sensor has the *V* metric set as '*Positive'*.

$$V_i^e = V_{ij}^u \qquad (7.3)$$

The formula 7.3 symbolises this option where a label of at least one negative *V* will result in the entire sensor being untrustworthy. Here the V_i^e represents the views of expert metric that is assigned to the entire sensor. V_i^e is calculated by extrapolating the rating of V_{ij}^u given to a sensor *i*, measurement *j* by user *u*.

Option B: The second option extrapolates the rating of the sensor measurement instances from the time frame of the actual user rating. Hence a label of at least one negative *V* will result in the sensor being untrustworthy from the time frame of that annotation.

$$\begin{gathered} f(V_{iy}^u) = Ratio\ (V_{iy}^u) \\ when\ f(V_{iy}^u) = 'Negative' \\ V_{ij}^e =' Negative' \ \forall\ j \geq y \\ V_{ij}^e = 'Positive' \ \forall\ j < y \end{gathered} \qquad (7.4)$$

The formulae 7.4 symbolises this option where the metric *V* is extrapolated from the time frame of the rated measurement *y*. A ratio obtained in case where conflicting ratings are provided by multiple users. Here the V_{ij}^e represents the views of expert metric that is assigned to all sensor measurements from time frame *j*. V_{ij}^e is calculated by extrapolating the rating of V_{iy}^u for all measurements where $j > y$.

7.2.2. Estimating Views of Expert Metric by Modelling Similarities

This method requires identifying patterns of other metrics of the sensor measurements that are already rated by users. Similarities based on these metric patterns can be used to label other measurement instances that do not have views of experts rating. For example, if a measurement instance has the views of expert rating as '*Negative'* has the pattern of the *H* metric as '1', *O* metric as '1' and *B* metric as '1' then metric instances with similar metric patterns can be labelled as '*Negative'*. By default a sensor has the *V* metric set as '*Positive'*.

$$V_{ij}^{e} = f(V_x^u) \;\; where, \; profile_x = profile_{ij}$$
$$f(V_x^u) = ratio(V_x^u \; in \; profile_x) \tag{7.5}$$

The formulae 7.5 estimates the user rating based on the profile of other metrics. Here the V_{ij}^{e} represents the estimated views of expert metric that is assigned to the sensor measurements j. V_{ij}^{e} is estimated by observing the similarities of its profile (profile_{ij}) with a profile ($profile_x$) of a rated (V_x^u) measurement. A ratio is obtained to manage similar profiles that have conflicting ratings.

7.2.3. The Inclusion of a Third State of 'Unknown'

This strategy assigns a default state of '*Unknown'* to the *V* metric to all sensor measurements that are not rated by the users. Ratings are not estimated or extrapolated in this strategy but are purely based on the user rating it as '*Positive'* or '*Negative'*.

Initially,

$$V_{ij} = 'Unknown' \tag{7.6}$$

Once rated,

$$V_{ij} = Ratio(V_{ij}^u)$$

Formulae 7.6 symbolises this method by initially assigning views of metric rating (V_{ij}) for the measurement as '*Unknown'*. Once the user ratings are available V_{ij} is set to the ratio of the ratings.

7.2.4. Incorporating Views of Expert Metrics with Trust Model

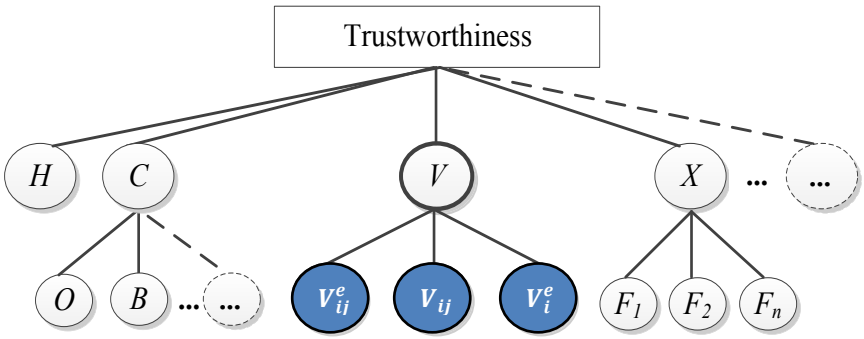


Figure 7.1: Incorporating the different views of expert metrics in the trust model

The above model (Figure 7.1) illustrates the representation of the views of expert (*V)* metrics V^e_{ij} (*V* metric extrapolated or estimated for the sensor measurement), V^e_i (*V* metric extrapolated to the sensor) and V_{ij} (*V* metric set to a default value) in trustworthiness model. Each type of *V* metric is grouped under the views of expert metric in the trust model to obtain a clear classification of the metric that are based on expert ratings. This representation has the advantage of natural classification when represented as an instance or individual of the trustworthiness ontology.

Clearly the trustworthiness model can be easily extended to represent multiple types of *V* metrics. Moreover in situations when multiple *V* metrics are available the user can decide on the metric to be included to assess trustworthiness. The experimental evaluation described in the next section demonstrates use of the different *V* metric types to assess sensor measurement trustworthiness.

7.3. Experimental Evaluation using the Views of Experts metric

The experiment investigates how the views of experts metric (*V*) can be used. Participants in the *UPLondon* (*UPLondon.org*) *Crackathon* event at the Urban Prototyping London (*sustainablesocietynetwork.net/th_event/crackathon*) festival in April 2013 were asked to rate 60 instances of sensor measurement windows. For these measurements they were provided with access to the full training data (sensor measurements), its contextual information (location of factories, streets, etc.) and visualization tools. They provided a rating for *V* of either Negative (non-trustworthy) or Positive (trustworthy) to each sensor window containing 100 measurements.

7.3.1. Experimental Overview

To evaluate the utility of the views of expert ratings, various experiments were conducted using the four options of calculating the *V* metric, that are extrapolating to entire sensor, extrapolating to sensor after the time of annotation, modelling the similarities and with the state of '*Unknown'*.

The 60 sensor measurement windows were taken from each of the untrustworthy sensors. The annotations provided by participants are used to calculate the *V* metric on other measurements. For example, when a sensor is annotated as 'Negative', its other measurement windows are annotated using these four *V* metric calculation options.

The same data set used for the experiment in chapter 5 containing 700 windows from trustworthy sensors and 300 windows from untrustworthy sensors is used for this experiment. Each experiment is run by changing the annotated sensor percentage (10%, 50%, 75% and 100%) for untrustworthy sensors. As an example, when using option *B* of the *V* metric calculation methods (section 7.2.1) if the annotated percentage of untrustworthy sensors is 10%, 28 windows of these sensors were annotated as 'Negative'. The remaining 272 measurement windows from untrustworthy sensors were rated as 'Positive'. Moreover when the annotated

percentage is 50%, 150, with 75%, 227 and with 100% 300 windows were annotated as 'Negative'.

The following summarises the options that are based on the methods described earlier to calculate the *V* metric for this evaluation:

Option 1: Views of experts metric extrapolated to sensor measurement using V_{ij}^{u} of already rated measurements.

Option 1.A: A label of at least one negative *V* will result in the entire sensor being untrustworthy.

Option 1.B: A label of at least one negative *V* will result in the sensor being untrustworthy from the time frame of that annotation.

Option 2: Views of experts metric assigned to sensor measurement by modelling profile similarities of rated measurements.

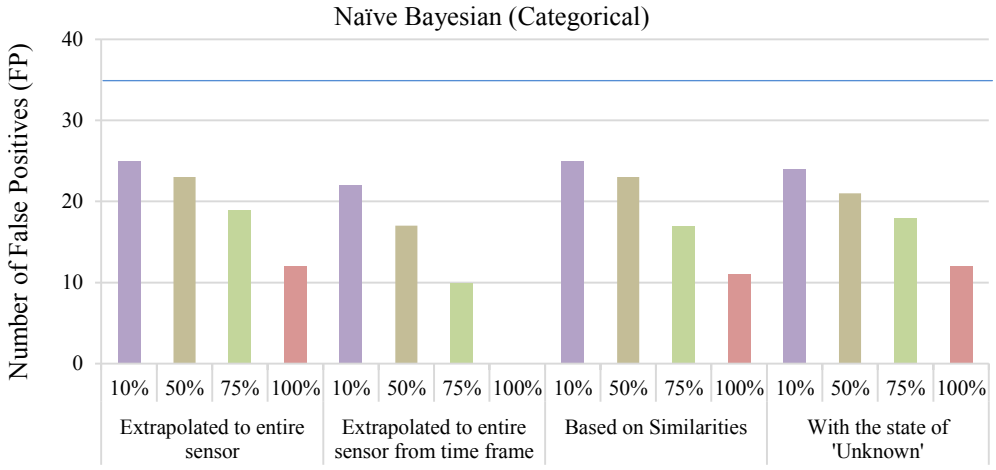
Option 3: A state of *'Unknown'* is given to each sensor measurement until rated by users as *'Positive'* or *'Negative'*.

The *V* metric values are inserted into the training data set and Bayesian models re-trained and the new models are used for prediction on the test data set. The term categorical is used here as opposed to binary (as in other chapters) due to option 3 of the estimation strategies resulting in more than two states for the *V* metric (Positive, Negative and Unknown). Continuous values can be obtained directly for options 1 and 2, however for option 3 a translation is used to convert the *'Unknown'* state to the value 0.5.

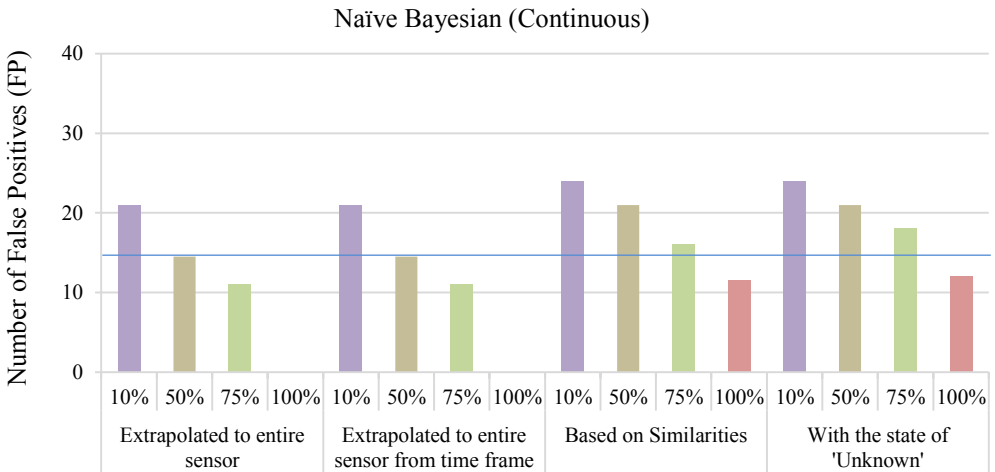
7.3.2. Comparing the Number of False Positives

All data displayed in the following graphs (Figure 7.2) are listed using confusion matrix included in the Appendix. The following graphs illustrate the number of *FP's* (False Positive) for the Bayesian models. The models include the views of expert metric (*V*) when testing the trustworthiness of the sensor measurements. For the convenience of comparison with the evaluation strategy without the use of the

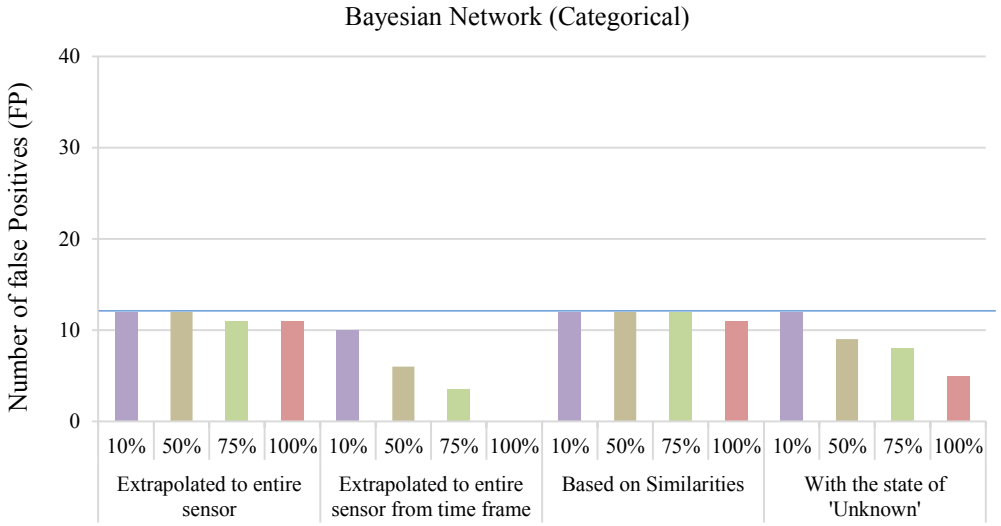
V metric (section 5.6) blue solid lines in the following graphs are used to indicate the number of False Positives.



(a)



(b)



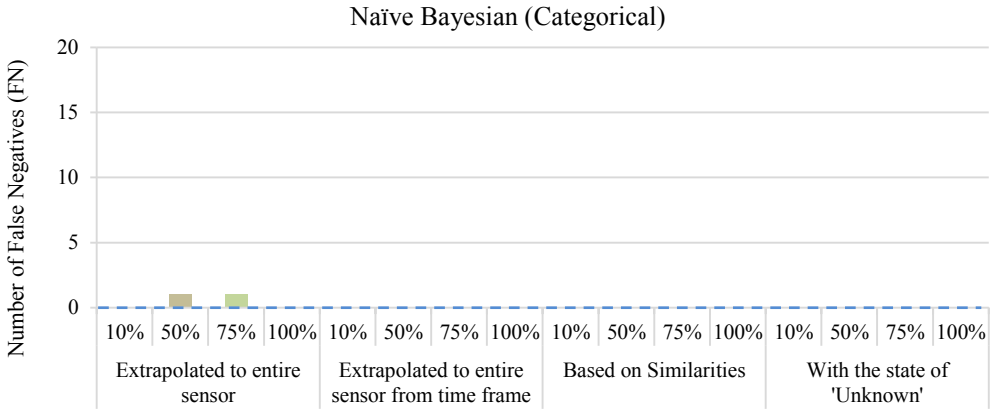
(c)

Figure 7.2: Number of FP's for (a) Naive Bayesian model with categorical data (b) Naïve Bayesian model with continuous data (c) Bayesian Network model with categorical data

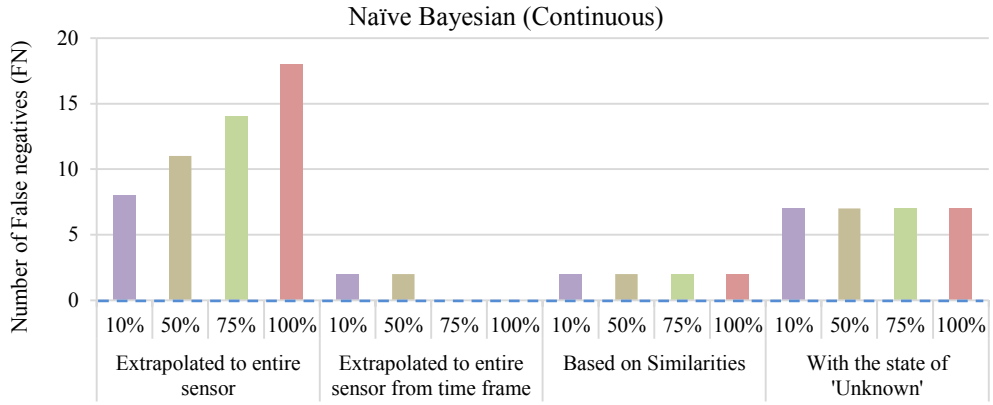
All Bayesian model strategies managed to improve on the number of *False Positive's* compared to the models without the use of the *V* metric. In particular the Naïve Bayesian and Bayesian Network models with categorical data have clearly improved when compared without the use of the *V* metric. However the Naïve Bayesian model with continuous data requires at least 75% measurement windows to be annotated in order to outperform the previous occasion that did not use the *V* metric.

7.3.3. Comparing the Number of False Negatives

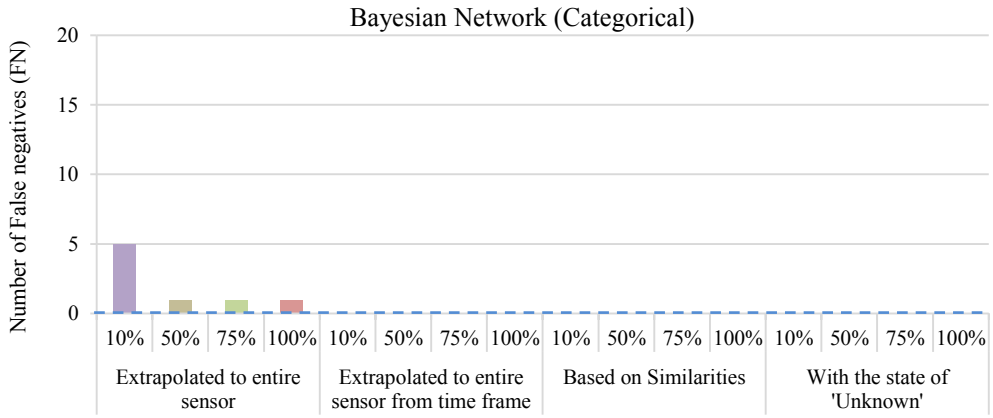
The following graphs (Figure 7.3) illustrate the number of *FN's* (False Negative) for the Naïve Bayesian and Bayesian Network model with categorical data and the Naïve Bayesian model with continuous data. The dotted lines in the following graphs indicate the *FN* rates for the Bayesian models without the *V* metric. In all cases the *FN* rate is 0 (section 5.6).



(a)



(b)



(c)

Figure 7.3: The number of FN's for the (a) Naive Bayesian model with categorical data (b) Naïve Bayesian model with continuous data (c) Bayesian Network with categorical data

All Bayesian models result in False Negatives (Figure 7.3) for the strategy when ratings are extrapolated to entire sensor. Furthermore the Naïve Bayesian model with continuous data produces false negatives for other options as well. This is due to the increase in sensitivity of the metrics. The increase in the sensitivity of the metric values when using continuous data as opposed to categorical is a compromise. While it can reduce the false positives it can also increase the false negative rate.

7.3.4. Analysis of Result and Comparing F1 scores

The results of the experiments clearly show that the Views of Experts increased the accuracy of the models with even 10% annotations resulting in an improvement in the accuracy in some cases (Figure 7.2.a). The results also demonstrate that with more Views of Experts available a higher accuracy was achieved. Moreover extrapolating the *V* based on certain strategies (e.g. extrapolating after time frame of rating) improved the accuracy when compared to other extrapolation options or when there was no extrapolation.

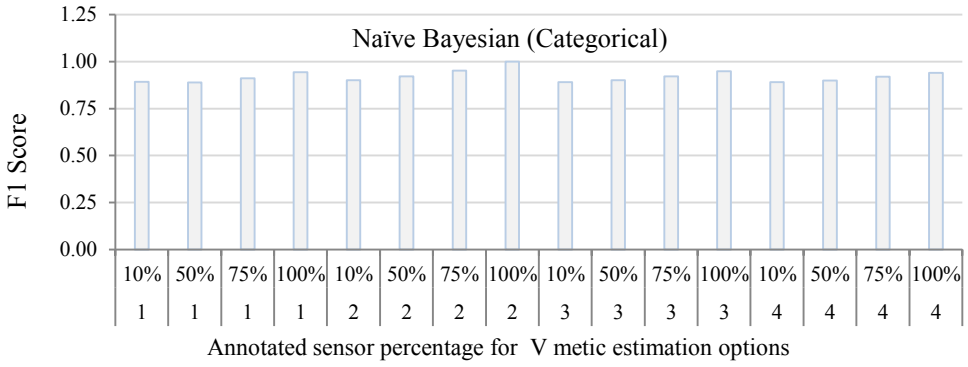
The Bayesian models with categorical data proved to be a good solution when the *V* metric is estimated using the options of extrapolating to sensor after the annotated time frame. Especially the Bayesian Network model provided the best results as it considered conditional dependencies. As seen by the results (increase in the false negatives in Figure 7.3), the option of extrapolating to the entire sensor has however suffered in accuracy comparatively. The rationale is that this option has resulted in the corruption of the training data. Moreover the increase of sensitivity of the metrics due to the use of continuous data also resulted in an increase of false negatives.

$$F1 = 2.\frac{precision\ .recall}{precision+recall} \tag{7.5}$$

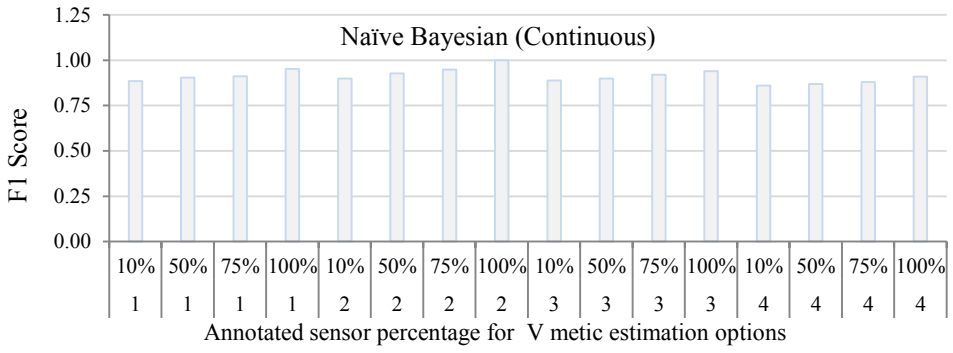
$$Precision = \frac{TP}{TP + FP} \tag{7.6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7.7}$$

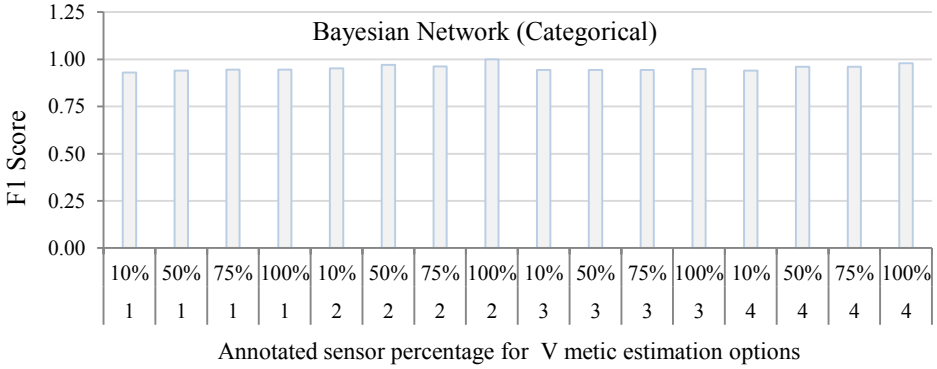
F1 scores are calculated for each *V* metric calculation methodology with varying types of available ratings. To summarise a ranking's value as a single number, the maximum *F1* score is used [117]. The *F1* score (formula 7.5) is a measure of a test's accuracy that considers the precision (formula 7.6) and the recall (formula 7.7) which are based on the True Positive (*TP*), False Negative (*FN*) and False Positive (*FP*) values of the test.



(a)



(b)



(c)

1 - Extrapolated to entire sensor
2 - Extrapolated to measurements from time frame of annotation
3 - Modeled on similarities
4 - With state of '*Unknown'*

Figure 7.4: The summary of F1 scores for (a) Naïve Bayesian with categorical data (b) Naïve Bayesian with continuous data (c) Bayesian Network with categorical data

The *F1* values for the experiments based on the trustworthiness determined by the different Bayesian models using *V* metric estimation methods are illustrated in Figure 7.4. The different methods of estimating the *V* metric is denoted from 1 to 4, followed by the percentage of available *V* metric (explicitly rated by the user) for sensor measurements. The *F1* scores for each experiment are close to the value 1, hence all tests are considered to have a high accuracy.

7.4. Related Work

Work by *Harper et al*. [118] explains the challenges of encouraging participation from the community to provide feedback on online content. Moreover the issue of under contribution is highlighted by *Resnik et al.* [47]. The strategies followed by *Harper et al.* and *Resnik et al.* focuses on motivating and endowing incentives for users to contribute towards such communities. However these do not address situations when the reviews or ratings are missing or inadequate.

The work in this thesis relates to the goals of *Marlin* [119] who describes a methodology based on a latent variable model to predict user ratings. Work by *Marlin* aims to predict the specific users prediction on a specific item but the requirement of WikiSensing is to estimate the trust rating in general by using all rated items (sensor measurements). *Cheung et al.* [120] describe an optimal user similarity function and user rating styles for memory-based collaborative recommender systems. They introduce a ratings transformation function into the ratings prediction formula in order to minimise a prediction error term. This work lays the foundation to enhance the way predictions are done that is based on

similarities of profiles. This can be a definite consideration to incorporate when estimating views of experts based on modelling similarities of other metrics. This will be further useful with the increase of other metrics that are calculated for sensor data.

7.5. Conclusion

This chapter discussed the methods to address the challenges when obtaining user ratings on large amounts of sensor data. These methods estimated or substituted values for missing data. The estimation and substitution were based on extrapolating values, observing existing metric patterns and using default values.

In these experiments, the accuracy of Bayesian models were improved by using additional metrics (e.g. views of experts). The availability of new evidence, e.g. views of experts, could clearly improve the models and their accuracy. As new observations become available, previous posterior data can be used as a prior. Exploiting this effectively would require the use of either a dynamic modelling approach or supporting the updating of models themselves. However it was also identified in certain cases (e.g. when ratings were extrapolated to entire sensor) additional data was counterproductive.

Overall the views of experts increased the accuracy of the models. As expected the availability of more annotations improved the models. The results indicate that the Bayesian models with categorical data proved to be better when using this metric as opposed to using continuous data that increased the sensitivity of metrics.

8. Modelling and Managing a Multilevel of Trust

WikiSensing's storage and trustworthiness model is explored with the intention of managing data that can produce a multilevel of trust. The advantage of having a multilevel of information is the ability to discover new information when data can be logically subdivided. This is a key motivation for calculating trust on different levels as it enables the cross validation of trustworthiness of a higher level from its lower levels.

A scenario on route data that is gathered for the visually handicapped is used throughout this section. This route data consists of a live collection of data streams from sensors such as accelerometers, gyroscopes and magnetometers. These sensors are maintained by the Bio-engineering department at Imperial College, London to generate a computer vision (annotated routes) to help the navigation of blind and partially sighted people in complex outdoor/indoor environments. This sensor data is managed by WikiSensing.

The example scenario presented contains route and segment traces recorded from a location at Imperial College, London to the main entrance of the London Science museum. The data on the sensors as well as the users responsible in taking the measurements are also maintained. A segment is a part of a route and a route can contain multiple segments. Routes are usually predefined by a route designer that includes a set of specific segments. Moreover a user can record several different routes using a sensor. This data can clearly be represented as a multilevel or a hierarchy of information and is used to check if the trustworthiness models can detect untrustworthy routes, segments, sensors and users.

8.1. Motivating Scenario

Mobility and transportation are considered one of the six dimensions of the smartness in *Smart Cities* [121]. *Hernández-Muñoz et al*. [122] state that sensors can be used to manage the mobility needs with appropriate intelligent transportation systems. Hence researchers have frequently worked on methods that enable members of the *BPS* (blind and partially-sighted) community to travel safely and independently in indoor and outdoor environments [123, 124].

The Bio-engineering department of Imperial College, London has implemented a system to capture navigational paths using sensors of mobile devices as well as measurement of Wi-Fi signals in buildings. These sensors also record contextual data that can affect the navigation of the *BPS* community (e.g. staircases, traffic lights, revolving doors). The data that is generated from these sensors are stored and queried using the WikiSensing API.

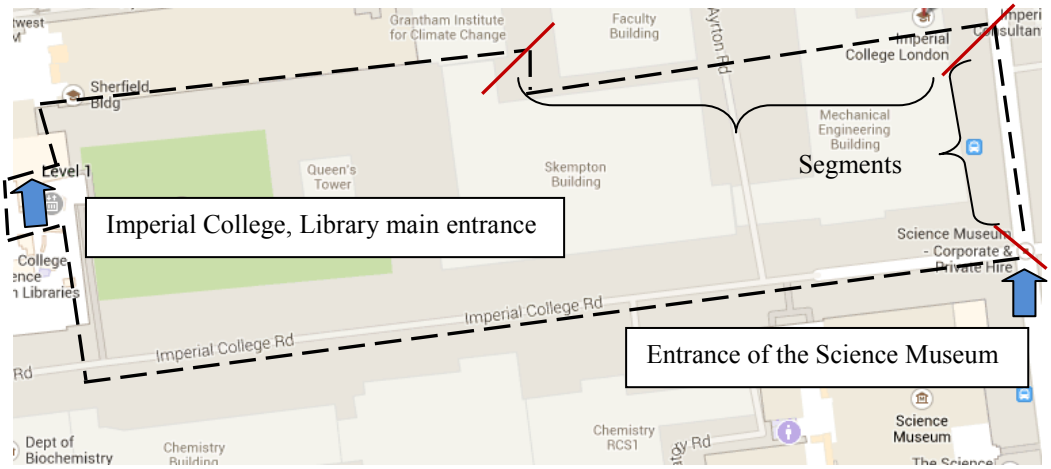


Figure 8.1: An example of a route instance for the trustworthiness assessment

Figure 8.1 illustrates a sample route taken between the main entrance of Imperial College, London library and the entrance of the London Science Museum. A route (represented using the dashed line) consists of a set of segments that are demarcated by the route designer. The route designer is a user who registers the

route with the navigational path system and is usually familiar with this route. These routes are recorded for the distance and the number of turns using an accelerometer and a gyroscope embedded in a mobile device.

Identifying the trustworthiness of these route instances is important as they are used to provide directions for the visually handicapped. A route recorded by a user is considered trustworthy when it contains the distance, the number of turns and the necessary contextual data (e.g. obstacles, revolving doors, staircases, etc.) that approximately corresponds to the actual route. The level of approximation can be based on the level of tolerance acceptable to the blind or partially sighted person. In contrast an untrustworthy route recording usually does not correspond with the actual route and may not be suitable for a visually handicapped person. To assess the trustworthiness, metrics can be calculated using the information on the route instances as well as any map information on the actual route.

The map information is generally a rough idea or an approximation on the actual route based on data such as the distance, the number of turns, etc. This information can be calculated using geographical coordinates, maps or can even be based on previously recorded routes that are correct. It must be noted that this map information itself is not sufficient to consider when checking for valid routes as they do not correspond to the current status of the route. For example, it may not contain certain obstacles or changes to the route that is relevant on a more up-to-date basis.

The inputs to assess the trustworthiness of routes are the map information and one or more route instances recorded by the users, with the problem being to attach a trustworthiness rating for each of these routes. Clearly the distance of a segment in route can be compared with the map information (Figure 8.2) but does not guarantee that it's trustworthy. Moreover the route can also be compared with other route instances to calculate similarities. Hence there is a need for new metrics and a methodology to calculate trust for these route instances. This contrasts with the trustworthiness calculations for environmental sensor data (discussed in chapter

5) due to the complexity of the route data which is based on instances rather than a continuous stream of measurements.

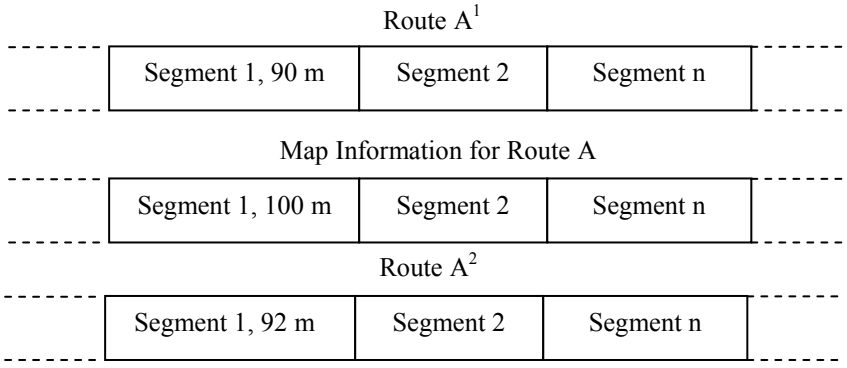


Figure 8.2: Comparing a segment with map information and other instances of that route

The trust metrics for the route data can include conflicts with background information (map information) by comparing the length and number of turns with known information. It can also include metrics on conflicts with other instances of the same route based on distance and number of turns. Moreover metrics for example, on the correlation of the segments of a particular route can provide a representation of the linear independence of this data. It can also include contextual information that can specifically affect the trustworthiness for a blind person as well as information based on the views of other users.

An important feature of the route data is the ability to represent it as a multilevel of information e.g. a sensor is used to record multiple routes, routes can be split into several segments, etc. The motivation of such representation for trustworthiness is that it enables a whole new dimension of data items that can be used to validate information. While a route instance can be assessed for trustworthiness, its segment instances can also be individually assessed for trustworthiness. Hence it is constructive to identify how the collective trustworthiness of segments can be used to validate trustworthiness of the route.

8.2. The Requirements (Challenges)

When trust is assessed based on multiple levels it is a challenge to compose or combine trust values of level *L* so that it is a correct reflection of the collective trustworthiness of level *L-1*. A multi-level of information exists when data can be subdivided into hierarchical levels. However not all types of data can have a multi-levelled structure (as discussed in section 8.5.2).

Figure 8.3 illustrates the different levels of information gathered when generating route data for the visually handicapped. Trustworthiness ratings can be calculated for users (*U*), sensors (*S*), routes (*R*) or segments (*G*). With a multilevel of information the number of trust metrics usually increases at lower levels as the data available can be divided into smaller components e.g. a route decomposed into a set of segments. Moreover certain contextual data becomes more relevant when information is partitioned, e.g. it is more appropriate to associate contextual data such as lifts and staircases with a certain part of a route (a segment) rather than the entire route.

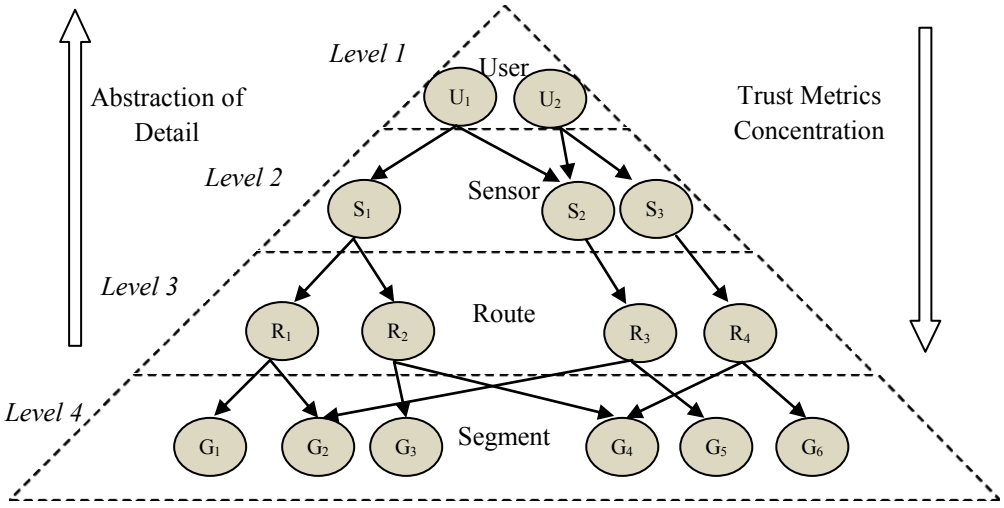


Figure 8.3: Multiple layered structure of trust in routes for visually handicapped

With the availability of trust metrics at different levels it must be possible to compose the trustworthiness (combine several trust ratings) of a certain level *L* to obtain the trustworthiness of level *L-1*. For instance, consider the division of a route into several segments. A methodology is needed to compose the trustworthiness for example, for segments G_1 and G_2 of Level 4 to determine the trustworthiness of route R_1 of Level 3. Hence how can we compose the trustworthiness of the segments of a route? Are conventional aggregation techniques or methods based on voting adequate for such composition? This becomes further challenging when segments have different levels of trustworthiness and is represented continuously as opposed to being discrete.

Consider the example scenario illustrated in Figure 8.4 that shows a set of route instances and the trustworthiness of its segments (in discrete and continuous form). For example, a trust rating of 0.5 or greater can be used as threshold to consider for a segment or a route to be trustworthy. Clearly if the trustworthiness of the route is composed by averaging or by voting the route instances B^1 and B^4 can be considered trustworthy and not-trustworthy respectively. However such methods may not be suitable for determining the trustworthiness of route instances B^2 and B^3 as they contain segments with very low trust ratings.

Route B^1

Segment 1, T (0.9)	Segment 2, T (0.8)	Segment 3, T (0.9)	Segment 4, T (0.7)

Route B^2

Segment 1, NT (0.3)	Segment 2, T (0.9)	Segment 3, T (1.0)	Segment 4, NT (0.4)

Route B^3

Segment 1, NT (0.1)	Segment 2, T (0.9)	Segment 3, T (1.0)	Segment 4, T (1.0)

Route B^4

Segment 1, NT (0.3)	Segment 2, NT (0.2)	Segment 3, NT (0.1)	Segment 4, NT (0.3)

Figure 8.4: Trust composition route example

With trustworthiness ratings available at multiple levels a querying mechanism is needed to search and aggregate information from different levels that match querying criteria.

8.3. Strategies for Composing Multilevel of Data

The problem is the need to compose the trustworthiness of certain levels so that it can correctly determine the trustworthiness of its higher level. The proposed solution is based on using the trustworthiness ratings of lower levels as metrics to determine the trustworthiness of the higher levels.

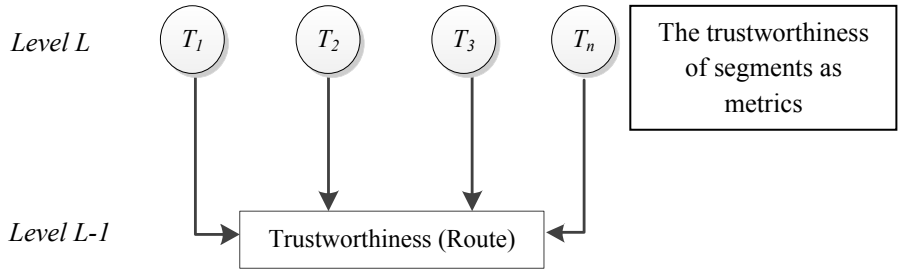


Figure 8.5: Assessing trustworthiness in a multilevel of information

To demonstrate this solution the trust ratings of the segments are used to determine the trustworthiness of the route. For example, the trust ratings of the segments can be used as metrics to train a Naïve Bayesian model (Figure 8.5) and to probabilistically determine trustworthiness ratings for route instances. This can be considered as a generic solution for trust composition however a certain degree of input from an expert may be required when considering contextual data that are available on routes (e.g. traffic lights, staircases, etc.). It must be also noted that a Naïve Bayesian model must be created for every route as the model may contain a varying number of nodes depending on the route. Due to the simplicity of the Naïve Bayesian model generating it for each route is not considered a noteworthy overhead. Moreover the Bayesian Network model is not useful in this case as there are no clear conditional dependencies between the trustworthiness of segments that impacts the trustworthiness of the route.

8.4. Trusting Annotated Routes for the Visually Handicapped

The route used for the trustworthiness assessment test starts from the main entrance of the Imperial College, South Kensington Campus Library to the main entrance of the Science Museum, London via Imperial College road and from the Science Museum entrance back to the library entrance via the Imperial College, Mechanical Engineering department. Figure 8.1 illustrates a Google map view of this route. The route contains ten segments and the contextual information is attached to it.

8.4.1. Example Route Data

Table 8.1 contains the route information and Table 8.2 contains the segment information that was recorded when obtaining data about the routes from the Imperial College, William Penny Building to the Science Museum, London. The data maintained are the *route id*, *sensor id*, *user* (responsible for the recording), *segment id, total distance, total number of turns* and the contextual data. A time stamp is also maintained as multiple recordings of the same route can be recorded by the same user.

Route Id	*Sensor Id*	*User*	*Time Stamp (Start)*	*Total Distance (Meters)*	*Total Number of Turns*
R_1	S_1	Jose	T_1	1423.74	5
R_1	S_2	Dilshan	T_1	1488.7	7
R_1	S_1	Nicola	T_2	740.2	2
R_1	S_2	David	T_3	2740.2	12

Table 8.1: Recorded sample instances for route R_1

Route Id	*Sensor Id*	*User*	*Time Stamp (Start)*	*Segment Id*	*Distance (Meters)*	*Total Number of Turns*	*Contextual Data*
R_1	S_1	Jose	T_1	G_1	324.78	0	
R_1	S_1	Jose	T_1+a	G_2	231.88	1	Revolving
R_1	S_1	Jose	T_2	G_3	113.3	3	
R_1	S_1	Jose	T_2+a	G_1	1.5	2	
R_1	S_1	Dilshan	T_1	G_1	337.62	1	

Table 8.2: Recorded sample segment instances in route R1

Table 8.3 and Table 8.4 contain the map information for the route and its segments that are used as background data for calculating metrics. This information contains a similar structure to the data recorded for the routes. However it also makes a distinction on turns based on the degree of its orientation.

Route Id	*Description*	*Distance (Meters)*	*Total Number of Turns*	*Number of Turns (between 0 and 90 degrees)*	*Number of Turns (90 + degrees)*
R_1	Imperial College, South Kensington Library to London Science Museum	1343	26	8	16

Table 8.3: Map information for route R_1

Segment Id	*Description*	*Distance (Meters)*	*Total Number of Turns*	*Number of Turns (between 0 and 90 degrees)*	*Number of Turns (90 + degrees)*
G_1	Library Entrance to Science Museum	337.20	5	1	4
G_2	Science Museum to Bessemer Entrance	222.34	3	1	2
G_3	Bessemer Entrance to William Penny Lab	159.10	2	1	1
G_4	William Penny Lab to Huxley Walkway Entrance	123.90	2	1	1
G_5	Huxley Walkway Entrance to Library Entrance	145.42	4	1	3
G_6	Library Entrance to Wolfson Room Printing Area	40.62	3	1	2
G_7	Wolfson Room Printing Area to Core Text Room (Einstein Bust)	38.43	2	0	2
G_8	Core Text Room to Fourth Floor Water Fountain	137.33	2	1	1
G_9	Fourth Floor Water Fountain to Second Floor	72.24	1	0	1

Segment Id	*Description*	*Distance (Meters)*	*Total Number of Turns*	*Number of Turns (between 0 and 90 degrees)*	*Number of Turns (90 + degrees)*
G_{10}	Second Floor Water Fountain to Library	65.51	2	1	1

Table 8.4: Map information for segments in route R_1

8.4.2. Managing a Multilevel of Data

The architecture shown in Figure 5.4 (Page 114) is used for managing route data. As the framework supports heterogeneous data, the multilevel of the route information can be managed without the need to modify or re-design the storage model. The route data recording are stored in the non-relational sensor database and Wiki pages generated for each route. Moreover the *Assess Trustworthiness* module of this architecture is extended to contain the functionalities to calculate the metrics for the route data. It obtains the route data and map information from the *sensor database* and records the metrics, calculations and trust rating in the *trust database*.

The trustworthiness including the calculated metrics is represented using an ontology similar to the one presented in section 5.4.2. The metrics for the multilevel information is represented using *OWL* classes and the hierarchical relationships represented using *object properties* and *sub classes*.

8.5. Examples of Applying the Trust Model

The definition of trustworthiness explained in section 5.2 states that, P *{User: Sensor.measurement(s) | E}* is the probability that the sensor provides a measurement(s) that is accepted by the *User* given such evidence. This chapter considers assessing the trustworthiness of the routes for the visually handicapped. Hence the definition is extended so that, it is the probability that the *Sensor*

provides a set of measurements (e.g. a route, segments) that is accepted by the *User* to be suitable for a visually handicapped person given such evidence.

Figure 8.6 illustrates the extended trustworthiness model that can be applied to a scenario of route data. The key distinction is that this model is applied to a multilevel of the data in contrast to flat data as demonstrated in chapter 5. For example, metrics are calculated for a route as well as for each segment that constitutes this route. Furthermore a new metric (*K*) on the correlation coefficient between instances of the same route is also included in the model. *K* calculates the correlation of the distance and the number of turns of the segments in the route. This metric is classified under the *C* (conflicts) class of metrics as it aims to identify conflicts based on correlation.

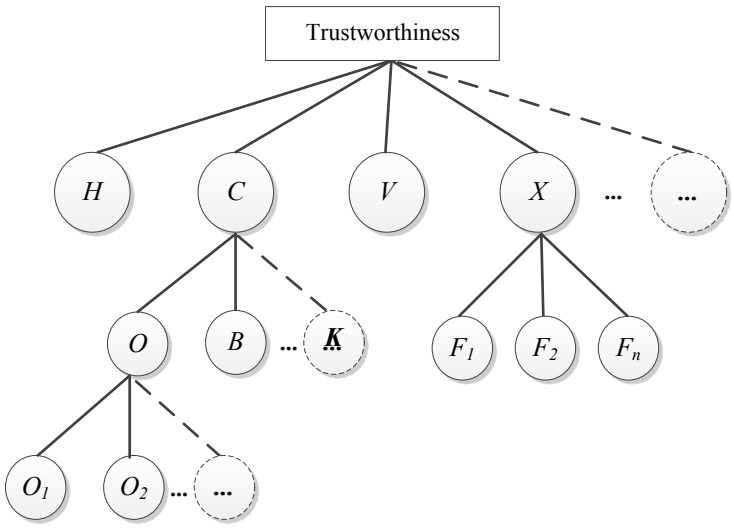


Figure 8.6: The extended trustworthiness model for route traces

Figure 8.7 shows how a multilevel of metrics is modelled. The B_L^{R1} metric represents the background information conflicts based on the length (subscript *L*) of route R_1. Moreover the metrics B_L^{G1}, B_L^{G2}, … B_L^{Gn} represent the conflicts based on the length of every segment of that route instance.

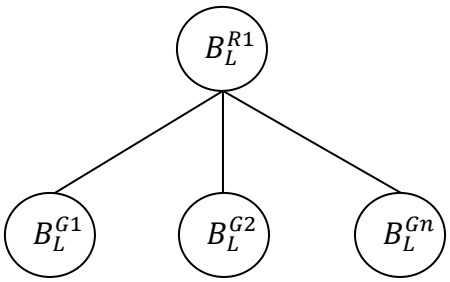


Figure 8.7: An example of a multilevel of metrics

8.5.1. Calculating the Metrics

The trustworthiness metrics are calculated for this domain of data based on multiple levels of the user, the sensor, the route and the segment. The superscript in the metric notation represents this level for example, *U* denotes the user, *S* denotes sensor, *R* denotes the Route and *G* denotes the Segment. The metrics are calculated for this example using default formulae as listed in Table 8.5.

The *H* metric (historical information) is calculated for users or sensors and is based on previous measurements that were recorded. At the user level, the *H* metric is the rating on trustworthy measurement that was previously recorded by a user. Moreover the historical data is modelled for routes (H_R^u) and for segments (H_G^u). Similarly at the sensor level, the previous measurements of the sensor are taken into account. The history of the sensor when measuring routes (H_R^s) and segments(H_G^s) can be calculated.

A richer set of metrics can be calculated at the level of the route. The metric that identifies conflicts with other instances of that route based on the distance (O_L^R) and the number of turns (O_T^R) is calculated. The Background information metric (*B*) is calculated by comparing the route instance with the map information. B_L^R is calculated by identifying if route is too short or too long and B_T^R is calculated by checking the total number of turns. The metric (*K*) is obtained for the route by calculating the correlation coefficient for the length (K_L^R) and the number of turns (K_T^R) in its segments with other instance of that route. In addition the *V* metric (views of experts) can be calculated from the trust ratings provided by users' on route instances.

Segments constitute a route, hence the metrics based on conflicts with other segment instances and background information can be also applied at this level. The conflicts with other segment instances on the length (O_L^G) and on the number of turns (O_T^G) is calculated. Moreover the segments can also be compared with its map information, e.g. for the length (B_L^G) and number of turns (B_T^G). The contextual factors (X) can be associated with segment for example; factors such as staircases, lifts, traffic lights, revolving doors, as well as permanent and temporary obstacles are taken into account in assessing the trustworthiness of segments.

Metric	FORMULATION DESCRIPTION
H_R, H_G	This metric calculates the trustworthy measurements rating for the historical information (formula 8.1). The historical information is based on previous routes and segments recorded by either a user or a sensor. m_i denotes the measurement for a route or a segment and w denotes the total number of instances. $$H = \left\{\sum_{i=1}^{w} f(m_i)\right\}/w \quad (8.1)$$ *Where,* $$f(m_i) = \begin{cases} 1 \;\; if \;\; (Trustworthiness(m_i) =' Not - Trustworthy') \\ 0 \qquad\qquad\qquad\qquad\qquad otherwise \end{cases}$$
O_L, O_T	This metric calculates the conflicts between different instances of the same route or segment traces by comparing the total distance (formula 8.2) and the total number of turns (formula 8.3) of that route or segment. The conflicts on the number of turns can be segregated into turns that are less than 90 degrees and turns that are greater than 90 degrees. This is subjected to a tolerance threshold β_O^L for the distance and β_O^T for the turns. The value of β can be based on user tolerance plus any additional threshold value. The number of compared routes or segments is denoted by k. $$O_L = \left\{\sum_{n=1}^{k} f(L_n, l)\right\}/k$$

Metric	FORMULATION DESCRIPTION
	(8.2) *Where,* $f(L_n, l) = \begin{cases} 1 & if(\lvert l - L_n \rvert > \beta_O^L) \\ 0 & otherwise \end{cases}$ $O_T = \left\{ \sum_{n=1}^{k} f(T_n, t) \right\} / k$ (8.3) *Where,* $f(T_n, t) = \begin{cases} 1 & if(\lvert t - T_n \rvert > \beta_O^T) \\ 0 & otherwise \end{cases}$
B_L, B_T	This metric calculates the rate of difference between the actual sensor measurement (a route or a segment) and the background data (map information). The metric is calculated for a route or a segment by comparing with the background information on the distance (formula 8.4) and the number of turns (formula 8.5). *l* and *t* denote the length and total number of turns of the route or the segment. *L* and *T* denote the length and total number of turns specified in the map information. The ranges are subjected to a tolerance threshold β_B. It is either added or subtracted based on whether *l* is greater than or less than *L*. $B_L = f(L, l) = \left\{ \frac{\lvert (l \pm \beta_B^L) - L \rvert}{L} \right.$ (8.4) $B_T = f(T, t) = \left\{ \frac{\lvert (t \pm \beta_B^T) - T \rvert}{T} \right.$ (8.5)
K_L, K_T	This metric calculates the correlation coefficient in segments for instances of the same route. The metrics K_L and K_T are calculated (represented as *K* in formula 8.6) for the length and the turns in segments for the route instance. R_x and R_y are the values of the route instances *X* and *Y* and M_x and M_y are the mean value. SS_X and SS_y are the squared deviation values for each route instance. The function $f(SS_X, SS_y)$ returns the correlation coefficient (*r*) of the specific routes. β_K specifies the correlation threshold.

Metric	FORMULATION DESCRIPTION
	$$K = \left\{ \sum_{n=1}^{k} f(r) \right\} / k$$ $$f(\mathrm{r}) = \begin{cases} 1 & if\ (r > \beta_K) \\ 0 & otherwise \end{cases} \quad (8.6)$$ $$\mathrm{SS}_X = \sum_{k=0}^{n} (X - M_X)^2$$ $$\mathrm{SS}_y = \sum_{k=0}^{n} (Y - M_y)^2$$ $$r = f(\mathrm{SS}_X, \mathrm{SS}_y) = \sum_{k=0}^{n} \frac{(R_x - M_x)(R_y - M_y)}{\sqrt{(SS_X \,.\, SS_Y)}}$$

Table 8.5: Metric Calculations Formulae for Route Traces

The contextual factors (X) for the route data can impact the trustworthiness either in a positive or negative manner. For instance, for the visually handicapped, a segment with a staircase may be less trustworthy when compared to an alternative segment with a lift. Moreover the β values (e.g. tolerance) used in the calculations are set manually based on an acceptance level of the user. Ideally it is a value that is set in accordance to what the user deems adequate based on their knowledge of the route.

8.5.2. Comparing Metrics for Route Traces with other Metrics

The goal of applying the trust model to sensor data is that it can be represented in a multilevel to understand the applicability of these models to determine trust in other data domains. Hence it is useful to identify how this type of data compares with flat structured sensor data discussed in previous chapters.

The trustworthiness model was successfully applied to environmental sensors that provide a sequential, flat data that are usually linearly ordered e.g. time stamp and sensor reading. In contrast the route trace data discussed in this chapter are spatial data that can be decomposed into further levels of information e.g. recording of routes decomposed into smaller segments. Furthermore the disposition of the environmental sensors were in most cases fixed (e.g. fixed pollution sensors) and the sensors to obtain route data are inherently mobile (e.g. accelerometers and gyroscope embedded in mobile devices). Additionally the recordings of environmental sensors are continuous numerical measurements as opposed to one-off geographical recordings of route traces.

The definition of trust in environmental sensors is the probability that a sensor produces a measurement that is accepted by the user to be correct. However considering the fact that these routes are intended to be used by the visually handicapped the definition is reformed to state that it is the probability that a sensor produces a trace of a route that is accepted by the user to be suitable for a visually handicapped person.

The difference in the structure of the data has changed the way the trust metrics are calculated. For instance, conflicts in environmental sensor data are based on numerical inconsistencies of measurement instances of spatially nearby sensors. Conversely in route data, conflicts occur when routes or segments are inconsistent (e.g. in length, orientation, etc.) with other instances of that route or segment. Moreover the historical data for environmental sensors consists of previous sensor readings. However due to the lack of continuity of the data (e.g. as in pollution sensors that continuously monitor environments), the historical information for routes can only be based on previous ratings (e.g. ratings on the calculated trustworthiness).

The background information for environmental sensors is usually valid data on the sensor or the deployed environment (e.g. minimum and maximum possible measurements of a particular location, etc.). For route traces this can be based on map information that is distances and orientations of segments or routes calculated

with the aid of maps. The contextual data discussed so far relates to environmental sensor. This information was used to reason on the trustworthiness of sensor measurement instances. For example, a measurement inconsistency due to the sensor not being calibrated resulted in the reduction of the trustworthiness. Moreover the trustworthiness was not affected when a measurement discrepancy occurred due to a nearby factory. On contrary, the contextual data for routes can be used to influence the trustworthiness of the route in a positive or negative manner. For example, obstacles such as staircases may diminish the trustworthiness of a specified route while a lift may increase it. Finally the sensors for obtaining route traces can be used by multiple users which were not evident in environmental sensors as they were usually owned or controlled by a single user.

8.5.3. Assess Trustworthiness on a Multilevel of Data

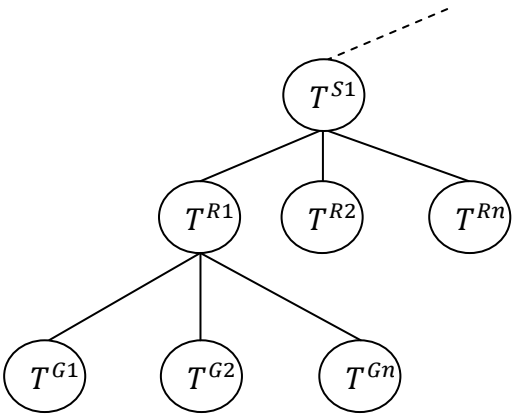


Figure 8.8: Trust calculated at multiple levels

The assessment of trustworthiness on multilevels of information results in multiple levels of trust metrics. These metrics can be used to assess the trustworthiness of that specific level (e.g. using the information from routes, segments, etc.). This differs from the trust assessment of environmental sensor data that has a flat structure.

Figure 8.8 shows trustworthiness calculated at several levels in line with availability of multilevel data. The trustworthiness of lower levels can be considered to be associated with the trustworthiness of its higher level. This hierarchical example illustrates how trustworthiness can be associated between segments and routes or between routes and sensors. This structure is also maintained when representing this information as an ontology.

8.6. Experimental Evaluation

The objective of the experiments is to evaluate the framework for trustworthiness detection on a domain of route data recorded for the *PBS* community. This contrasts with the previous evaluation (chapters 5 and 6) as data that can be represented as a multilevel of information. To do this, untrustworthy routes are deliberately recorded and are injected into a set of route data to check whether the models and tools are sufficient. The Naïve Bayesian and Bayesian Network models are compared in calculating the trustworthiness.

8.6.1. Experimental Overview, Data Sets and Parameters

The trustworthiness framework for multilevel of data is evaluated using three categories of route data that is based on trustworthiness. The *first category* contains trustworthy route and segment instances that are obtained with caution. The *second category* contains non-trustworthy route and segment instances that include deliberate errors (e.g. very long or short routes, wrong number of turns, etc.). The *third category* contains non-trustworthy route instances that include only a few (a maximum of 4 out of 10 segments) non-trustworthy segment instance. This category demonstrates route instances that may seem to be trustworthy when considered as a whole but are not trustworthy as it contains untrustworthy segments. Figure 8.9 exemplifies this category with route instances R_2 having a total distance comparable with the trustworthy route instance of R_1. However the

route should not be considered trustworthy as it clearly contains untrustworthy segments (segments 1 and 2).

Route R_1, (Trustworthy), Total Distance = 770m

Segment 1, 100m	Segment 2, 300m	Segment 3, 150m	Segment 4, 210m

Route R_2, (?), Total Distance = 767m

Segment 1, 25m	Segment 2, 380m	Segment 3, 152m	Segment 4, 210m

Figure 8.9: Example route instance for category three

The data set used in this experiment contains 100 route instances recorded from the main entrance of *Imperial College, London Library* to the main entrance of the *London Science Museum*. Each route instance contains 10 segments. The 100 route instances contained 54 trustworthy and 46 untrustworthy route instances. Out of the 46 untrustworthy route instances 27 belong to category 2 and 19 belong to category 3. These route instances are randomly split into training data and test data. The training data contains 40 trustworthy route instances and 30 untrustworthy route instances with 18 instances for category 2 and 12 for category 3. The test data contains 14 trustworthy route instances and 16 untrustworthy instances with 9 instances for category 2 and 7 for category 3 (summarised in Table 8.6).

Category	*T*	*NT*
1	54	0
2	0	27
3	0	19
Total	54	46

	T	*NT*	
Training	40	30	
Category (2, 3)	0	18	12
Test	14	16	
Category (2, 3)	0	9	7

Table 8.6: Breakdown of experimental data for routes

The Metrics are calculated for each route and segment instances. The tolerance threshold values of β_O^L , β_O^T, β_B^L and β_B^T are set to 0 in order to achieve a higher level of sensitivity. The different Bayesian model strategies are compared by applying the metrics generated for the route data. For this example the contextual data is not used to train the models but instead can be used to annotate

the determined trustworthiness. This is due to the contextual factors in route not impacting the actual recording of that route as with pollution sensor data (e.g. a nearby factory causing sensor measurements conflict). Moreover it is an indication to the user that in addition to the trustworthiness rating of the segment the contextual data can also be considered.

8.6.2. Assess Trustworthiness of Routes

This experiment assesses the trustworthiness of route instances using metrics based on the total length and the total number of turns. The following feature vector (Table 8.7) shows a sample set of the calculated metric values and trustworthiness for route instances (*RI*). Column *B* represents the metrics calculated using map information and *O* represents the metrics calculated by comparing other route instances. Metrics are calculated based on the distance and the number of turns.

RI	*B*		*O*		*T*
	B_L^R	B_T^R	O_L^R	O_T^R	
1	0.01	0.0	0.32	0.37	Y
2	0.03	0.27	0.42	0.47	Y
12	0.58	0.73	1.0	0.84	N
13	0.6	0.87	1.0	0.95	N
16	0.03	0.07	0.32	0.32	N
16	0.03	0.33	0.19	0.42	N

Table 8.7: A feature vector of a sample set of training data for routes

		Predicted				
	Naïve Bayesian (Binary)		Naïve Bayesian (Continuous)		Bayesian Network (Binary)	
Actual	T	N	T	N	T	N
T	TP 14	FN 0	TP 14	FN 0	TP 14	FN 0
N	FP 7	TN 9	FP 7	TN 9	FP 7	TN 9

Figure 8.10: The confusion matrix for test route data for Bayesian model strategies

Figure 8.10 and Figure 8.11 illustrates the entire outcome for the Bayesian model strategies. A high number of false positives are generated for the route data for all Bayesian model strategies. All models fail to detect the untrustworthy routes in category 3 (containing non-trustworthy route instances with only a few non-trustworthy segments) as the metrics do not detect any problem with the length of the route or the total number of turns.

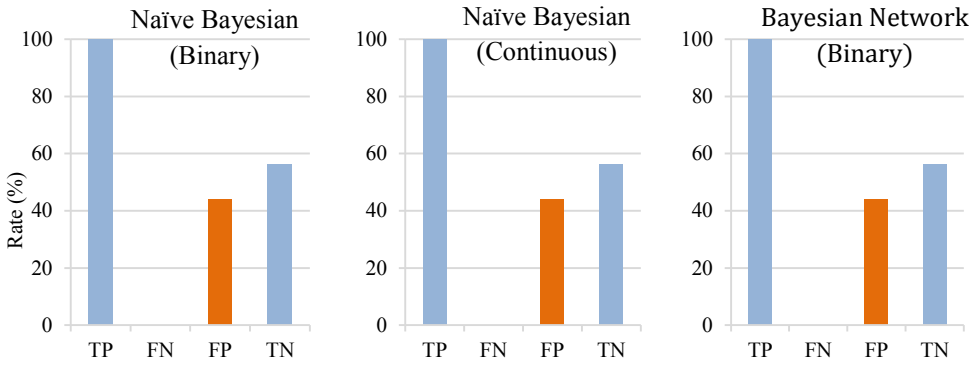


Figure 8.11: Summary of results (percentages) for test data

The three approaches resulting in the same outcome is mainly due to limitations of available training and test data. Data on a single route was available for this experiment that caused metric values to be limited within a specific range. This has clearly impacted the training of the models and reflected on the outcome to be uniform across all three strategies.

8.6.3. Inclusion of Correlation Coefficient Metric

This experiment is similar to the previous, but includes the *K* metric (correlation coefficient) for trustworthiness assessment. The *K* metric is calculated for all 100 routes based on its segments as described in formulae 8.6. The correlation coefficient is calculated between the segments of the route instance and the segments of the map information. *Cohen* [125] states that a correlation of 0.5 or greater is large and anything less than that is either moderate or small. Hence the correlation threshold β_K is set to 0.5 to consider a good correlation between the

route and the data from map information when converting the *K* metric from real values to binary.

Figure 8.12 and Figure 8.13 illustrates the outcomes for the Bayesian model strategies. The false positives in experiment 1 were for category 3 (route with a subset of untrustworthy segments). It failed to correctly detect these route instances as the metrics (based on the total distance and the number of turns) do not detect the problems with the untrustworthy segments. However when the *K* metric (correlation coefficient) is incorporated in experiment 2 this category of untrustworthy route instances are detected. This is due to the *K* metric identifying the strength and direction of the linear relationship between the recorded segments and the map information.

		Predicted				
	Naïve Bayesian (Binary)		Naïve Bayesian (Continuous)		Bayesian Network (Binary)	
Actual	T	N	T	N	T	N
T	TP 14	FN 0	TP 14	FN 0	TP 14	FN 0
N	FP 0	TN 16	FP 2	TN 14	FP 0	TN 16

Figure 8.12: The confusion matrix for test route data with the correlation metric (*K*) for Bayesian model strategies

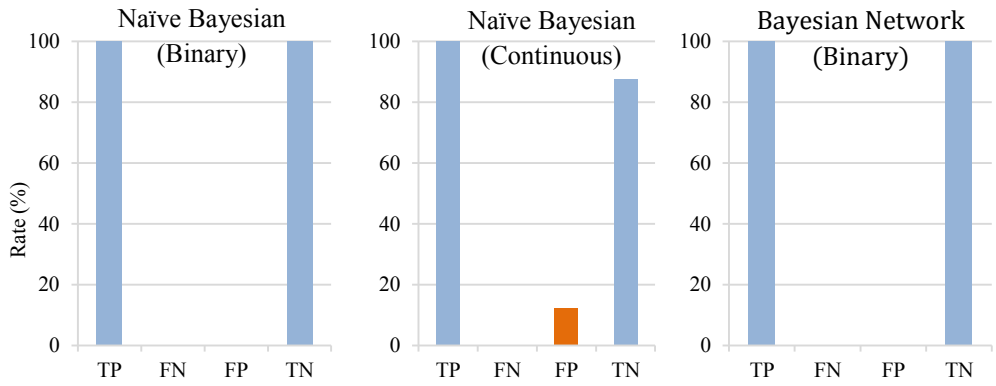


Figure 8.13: Summary of results (percentages) for test data

Hence a metric such as a *correlation coefficient* is required to detect the inconsistencies of segments when the route is assessed for trustworthiness. In conclusion when assessing the trustworthiness of higher levels in a multilevel of data, metrics are required to detect problems of data in lower levels as well.

8.6.4. Composing Trustworthiness Values of Segments

The objective of this experiment is to compose trust values of segments in order to determine the trustworthiness of the route. The first stage of the experiment involves determining the trust values of the segments and the second stage involves using the trust ratings generated from the first step to compose the trustworthiness of the route.

RI	*GI*	*B*		*O*		*T*
		B_L^G	B_T^G	O_L^G	O_T^G	
1	1	0.04	0.0	0.68	0.47	Y
1	2	0.04	0.5	0.68	0.37	Y
12	1	0.88	1.0	0.84	0.37	N
12	2	1.0	0.5	0.84	0.37	N
16	1	0.77	1.0	0.95	0.37	N
16	2	0.52	1.0	0.89	0.63	N

Table 8.8: A feature vector of a sample set of training data for segments

The original dataset contained 100 routes with 10 segments each. Moreover the metrics are calculated for these 1000 segments based on the total length, the total number of turns. Out of the 1000 segments are approximately 700 segments are trustworthy and the remaining untrustworthy. A sample set of training data is shown in the following feature vector (Table 8.8). The first column *RI* is the route instances and *GI* is the segment instances and is followed by the calculated metrics. Similar to the previous experiments the data is split into training (700) and testing (300). The Naïve Bayesian models (binary and continuous) are used to determine the trust values of segments. This model is used due to its simplicity and the fact

that no significant conditional dependencies were identified. Trust ratings for 300 test segments and the 700 training segments are the output of the first stage of the experiment. The objective is to obtain some trustworthiness ratings by applying the trust model. These trust ratings are used for the next stage of this experiment.

The second stage of the experiment uses Naïve Bayesian models that are trained with trustworthiness ratings (binary and continuous) resulted from the first stage. The selection of the data set is the same as the configuation in section 8.6.1 with the 100 route instances contained 54 trustworthy and 46 untrustworthy route instances with 30 instances used in testing. However instead of the conventional metrics that were based on distance, number of turns, etc. this data set contains the actual trust rating for the segments. Moreover conventional aggragation methods of voting and averaging is also used to compose the trustworthiness of these segmenets.

RI	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8	G_9	G_{10}	*Expected T*
R_1	0.9	0.9	0.9	1.0	1.0	0.9	0.9	0.87	1.0	1.0	T
R_2	1.0	0.9	0.7	1.0	1.0	0.9	1.0	1.0	1.0	1.0	T
R_3	1.0	0.8	0.6	1.0	1.0	0.9	1.0	1.0	1.0	1.0	T
R_4	0.0	0.0	0.4	0.8	1.0	0.2	1.0	1.0	1.0	1.0	NT
R_5	0.0	0.9	0.0	1.0	1.0	0.9	1.0	1.0	1.0	1.0	NT
R_6	0.0	0.0	0.2	0.0	0.8	0.0	0.0	0.08	0.0	0.0	NT

Table 8.9: A sample set of test data

Table 8.9 illustrates a sample set of test data containing trust ratings of segments for trustworthy and non-trustworthy route instances. For example, cell R_1, G_1 represents the trustworthiness rating for segment 1 of route 1. This data is used to compose the trustworthiness of segments using Bayesian models and the aggregation methods of averaging and voting. For example, a method based on voting or averaging fails to detect the untrustworthy route instances R_4 and R_5. However the Bayesian models have correctly determined these instances.

	Predicted			
	Naïve Bayesian (Binary)		Bayesian Network (Continuous)	
Actual	T	N	T	N
T	TP 14	FN 0	TP 14	FN 0
N	FP 0	TN 16	FP 0	TN 16

(a)

	Predicted			
	Averaging		Voting	
Actual	T	N	T	N
T	TP 14	FN 0	TP 14	FN 0
N	FP 7	TN 9	FP 7	TN 9

(b)

Figure 8.14: The confusion matrix for trustworthiness of segment data for (a) Naïve Bayesian models (b) aggregation methods

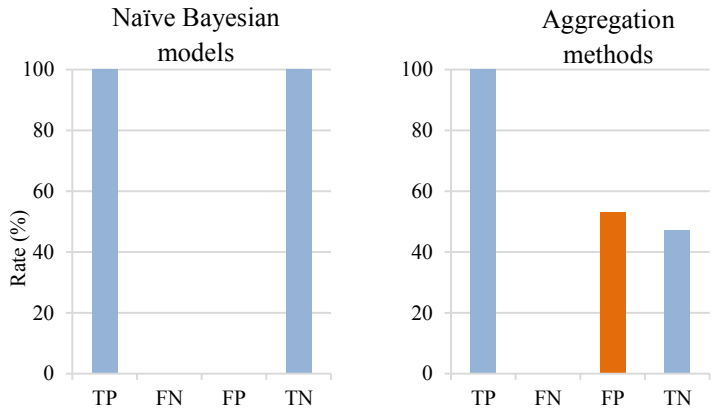


Figure 8.15: Summary of results (percentages) for test data

The results in Figure 8.14 and the summary in Figure 8.15 shows that using probabilistic Bayesian modelling is a good way to compose the trustworthiness of the segments when compared to an aggregation method such as averaging or voting. Clearly a statistical classifier such as Bayesian modelling is suitable to compose data as it considers evidential probabilities.

8.7. Conclusion

The extended trust model was able to assess the trustworthiness of route data represented at multiple levels. The multilevel of data imposed the challenge of trust composition. The experiments showed that metrics were needed to detect the linear relationships of lower levels when assessing trust in a higher level to reduce the number of false positives. Another solution is to segregate the turn counts by grouping them based on the degree of change in direction. This provides useful information if the turn involved in a change of direction (e.g. turns greater than 90 degrees) or if it is a mere change in trajectory. Metrics based on this information can provide further information on route data that was not possible when all turns were grouped together. Bayesian modelling also provided better results when composing the trust rating of segment instances when compared with conventional aggregation methods such as averaging or voting.

A possible next step includes applying the models to data taken from different routes. The objective is to gather routes with varying number of segments and to check models can correctly determine trustworthiness. It is also planned to further investigating the use of contextual data for routes when assessing trustworthiness. This would require understanding how they relate to the trustworthiness of a route from the perspective of visually handicapped people.

The work by *Petrie et al.* [126] describes a travel aid (known as *MOBIC*) to increase the independent mobility of blind and elderly travellers using Geographical data. Similar to the map information used in this thesis the *MOBIC* system calculates route information using digitised maps. Moreover they also highlight the importance of contextual data in order to augment the information of a route. However their approach does not assess trustworthiness for these routes and it will be interesting to investigate how the trust framework and models can be used with data from such systems.

9. Summary, Conclusion and Future work

9.1. Summary and Contributions

9.1.1. Summary

This research has introduced a new collaborative approach for sensor data management known as WikiSensing. The thesis presented an architectural design and described the implementation details for a collaborative sensor data management system. The advantage of WikiSensing is based on incorporating online collaboration into sensor data management. Online collaboration is used to annotate, update and share sensor information as well as in creating virtual sensors. The concept of virtual sensors is an extremely useful feature that provides sensor readings using existing sensor data streams. Some of the main challenges in sensor data management with online collaboration are due to the large amounts of heterogeneous, real time of sensor data as well as the need to demonstrate trust of the shared information.

This research investigated the challenges of managing trustworthiness in collaborative sensor systems. A framework and methodology was presented based on a generic probabilistic definition of trust, and described how to capture and calculate metrics for different types of available evidence. Trustworthiness was defined as a probability as it provides a good indication of uncertainty as well as to be used in future predictions. The approach is extensible allowing incorporating metrics based on other probabilistic models if needed, e.g. by using binomial models to calculate trust based on historical interactions with individual sensors as in other work [63]. A number of experiments were also presented to demonstrate and verify the use of the framework and models. Furthermore different representational models were compared and studied as to how early untrustworthy behaviour of sensors could be detected.

The WikiSensing system was one of the data stores that supported the *Hackathon* event at the *Urban Prototyping London* (*uplondon.org*) festival in April 2013. During the event, a workshop was held where contestants were given air pollution data similar to the one used in this thesis. They used WikiSensing's anomaly and conflict detection tools to obtain different metrics and to assign their own trustworthiness scores to sensor measurements at different time frames. They were also provided with various visualization tools to explore such data.

The practical experience from the workshop not only provided valuable feedback but also highlighted the opportunity for developing new metrics to extend the trust model easily. For example, it is not difficult to include metrics based on correlation values [127] between sensor measurements across different time periods (as shown in the metric calculation for the route data in section 8.5). It is also not difficult to include ones that explore the evolution of trust over time for individual sensors or those that capture trust propagation information [54] between different sensors. Although guidance was provided in compiling threshold values of β for metric calculations it has to be explicitly set by the user, based on discretion and knowledge. In future it is preferable to automate the estimates of β values taking into account available sensor information and previous trust scores.

9.1.2. Contributions

In this thesis, the architecture and a system for collaborative sensor data management as well as a model and a framework for trustworthiness management were presented. The central problems that this thesis concerns can be summarised as follows:

> *To find a means for managing collaborative sensor data (Big data) and to standardise the ways users can trust data in such environments.*

The research work presented here covers a range of different areas pertaining to the data management of sensor data, the organising of collaborative information and

the management and assessment of trustworthiness of the sensor data. The contributions of this research can be categorised as follows:

Collaborative Sensor Data Management: Managing sensor data and organising collaborative information with the aim of addressing the Big data challenges of volume, velocity and variety.

Trustworthiness Management: Managing and assessing the trustworthiness of sensor data with the aim of addressing the Big data challenge of veracity.

The key contributions of this research are based on the WikiSensing system that provides collaborative sensor data management. Furthermore it also provides trustworthiness management with the ability to assess trust on a multilevel of information. The contributions are summarised as follows:

An architecture and implementation of a collaborative sensor data management system known as WikiSensing

The distinct features of WikiSensing include a hybrid data storage, support for online collaboration and virtual sensors. The system is also used as a testbed to develop a framework to manage trustworthiness of sensor data based on a novel probabilistic model.

A generic probabilistic definition of trust in sensor data

A generic mathematical definition is provided to relate the general concept of trustworthiness with trustworthiness of sensor data.

A framework and model to determine the trustworthiness of data in a collaborative sensor data management system

The thesis describes a framework to capture, calculate and represent the metrics needed to determine the trustworthiness. The methodology of determining the trustworthiness is based on *Bayesian* probabilistic modelling. This work also describes the architecture and implementation of this framework based on a software system in order to implement the methodology.

An adaptation of the trustworthiness model when data that can be represented in a multilevel

The trust model is extended to assess trustworthiness when data is represented in a multilevel of information. The aim is to use this methodology as a generic solution to determine trust of data in other collaborative sensor data domains.

9.2. Current Applications of WikiSensing

There are several applications that make use of data and services of WikiSensing. One such application is the analytical workflow system known as *Concinnity* that benefits from the openness of WikiSensing. Furthermore other applications use the functionality of virtual sensors, uses it as a part of an elastic sensor information management system and as a cloud based informatics platform.

9.2.1. The Concinnity Platform

WikiSensing is the data management layer of the workflow system known as *Concinnity* [128, 129]. The main goal of *Concinnity* is to build a *WikiModelling* workflow facility on top of WikiSensing. It enables rapid development of applications built on sensor data using data fusion and composition of models to form novel collaborative workflows. The *WikiModelling* system consists of an *AppEditor* to model workflows and a *WorkflowEngine* to process the modelled workflows with the aid of WikiSensing.

The *AppEditor* supports developers in constructing sensor data applications by allowing them to retrieve sensor data from various sources using a declarative query language. Workflow definitions from the *AppEditor* are passed to the workflow engine which is used to retrieve the heterogeneous data from WikiSensing and filter or fuse it as required. It supports a wide variety of input and output data sources including wikisensing.org and requires that every model or data fusion module to register its list of inputs and outputs to the engine. A plug-in

architecture supports multiple methods for achieving data fusion and resolving concerns with trustworthiness. The following examples on health and medical applications and charge grids for electrical cars demonstrate how these applications use the functionalities of WikiSensing.

The aim of this project is to understand the future impact of the electrical charge grids needed for electric vehicles to a city's infrastructure. The *London's Digital City Exchange* has proposed a collaborative approach [130] that involves experts in transportation and electrical grids to model the impact of electric vehicle charging upon the electricity grid. The aim is to use an agent based transportation model to simulate expected journeys within the cities. WikiSensing *REST* services are used to manage the journey information generated from these simulations consisting of states of charge of the electric vehicles and their locations through the day. A workflow is created using the *AppEditor* of the *Concinnity* platform, and is driven using data from a range of sources including *National Statistics,* maps and electricity grid statistics. The information outputted from this workflow is again stored in WikiSensing.

Smart phones nowadays have many embedded sensors ranging from microphones to gyroscopes and proximity sensors. Similarly, new generations of professional wearable medical sensors can now connect to smart phones and transfer sensing results directly about person's health (e.g. blood pressure, oxygen saturation, blood glucose level, electrocardiogram, etc.). However, the lack of standard formats of storing and exchanging this data has created heterogeneity and disparity challenges, making it difficult for users to reclaim back their data, manage or remix it in their preferred ways. From the provider's point of view, such massive growth of these big health sensor data creates both data manageability and collaboration challenges. For example, an application that analyses the functional magnetic resonance imaging (*fMRI*) collects a sequence of brain images in order to localize brain activities that rely on neuron activity across the brain or in a specific region. These detected activities have proved to be useful to plan for surgery and radiation therapy of the brain [131]. The data produced in such activities are

usually very large and are heterogeneous in nature. Moreover collaborative efforts are required to curate such large amounts of data. WikiSensing provides solutions for these problems by providing an efficient heterogeneous data management plus a platform to support collaboration. It allows individuals to contribute data about their lifestyles and well-being via smartphones, tablets and wearable devices (analogous to sensor devices). Furthermore WikiSensing enables the collected data to be available for sharing and collaboration.

Overall, WikiSensing provides a simple and accessible sensor data storage solution. One of the key advantages of WikiSensing noted by the users was the ability to dynamically set the data stored in each sensor point adding and removing fields as required. This schema-less design benefits rapid prototyping and the ability to support image storage is also useful. Coupled with this the custom search API provides a useful data access mechanism for retrieving data. According to the users the main challenge was based on the limited cryptographic security of data. Moreover they list several potential enhancements for WikiSensing. Most notable would be the ability to search for public data sets; this however would depend on some ontology being applied to each sensor which remains a key research challenge. Further enhancements to the API offering would expand the potential uses for the service. Such enhancements would include further trustworthiness assessment API's, an implementation of virtual sensors and other statistical assessment routines for sensor data.

9.2.2. Virtual Sensors based on Trustworthiness

Creating virtual sensors as explained in chapter 4 involves the composition of multiple sensor data streams into a single stream. The considerations taken were based on the types of the sensors and spatial distances of the contributing streams relative to the location of the virtual sensor.

The conventional virtual sensors (without considering trustworthiness) may contain the aggregation of sensor data streams that may not be trustworthy. If

such untrustworthy sensor streams are included the trustworthiness of the composed virtual sensor may be jeopardised. Moreover when virtual sensors are created over other virtual sensors these untrustworthy sensor streams can have a further cascading impact. The use of trustworthiness management framework enabled the assessment of trust for a sensor or a sensor measurement. Hence considering the sensor's trust rating when composing several data streams can enhance the quality (based on the trustworthiness) of the virtual sensor.

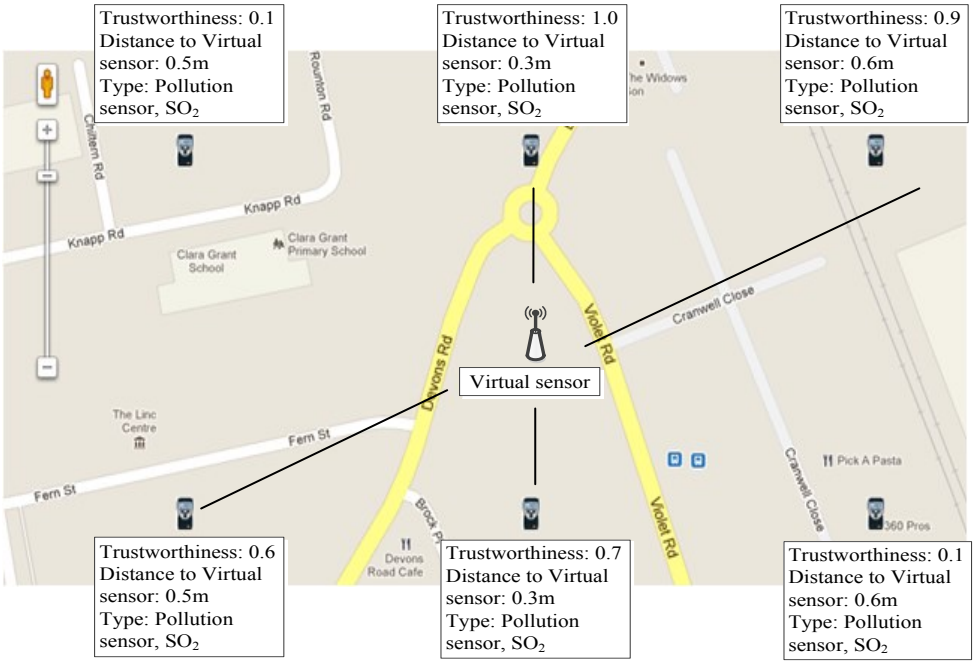


Figure 9.1: Attributes to consider when creating virtual sensors

The trustworthiness of the sensor can be used to decide on what streams are selected to compose the virtual sensor. Moreover if the trustworthiness is provided as a numerical value (a probability) it can also be used as weighting when composing the virtual sensor stream. Figure 9.1 illustrates an example of a virtual sensor created by selecting sensors with a trustworthiness rating over 0.5.

$$Virtual\ Sensor\ measuremet\ = \frac{\sum_{i=1}^{n} f(S_i) * T_i}{n} \qquad (9.1)$$

In addition to selecting sensors based on their trust rating the aggregated sensor measurements can also be subjected to weightings. This weighting can be applied in conjunction with other parameters such as weightings based on distance. The formula 9.1 symbolises an example of an aggregated measurement based on weightings of the trust rating. Here $f(S_i)$ represents any aggregate function that composes the data streams of the selected sensors (*1 to n*). Each aggregated value is subjected to the trustworthiness value $T_i$of the sensor.

9.2.3. EIMAP Monitoring in Large-scale M2M Sensor Networks

Managing urban air pollution monitoring applications in large-scale machine to machine (*M2M*) sensor networks require information management over widely distributed sensors under restricted power, processing, storage and communication resources. Elastic Resource Allocation strategies are novel management techniques based on Elastic Computing that can be used to address these challenges.

The *EIMAP* (Elastic Information Management for Air Pollution) by *Ma et al* [132] is a real-time air pollution monitoring system with high-performance information management in an elastic manner. This system has a four-layer architecture which contains thousands of sensors distributed over an urban area to monitor airborne pollutants. The potential data volume that is processed by this system varies from several bytes (e.g. individual readings per sensor) to the range of 8GB (e.g. whole readings per sensor per day that are used to capture high-resolution urban air pollution distribution resulting from transportation).

WikiSensing is used to simulate the lower two layers of the *EIMAP* system and to evaluate the capability of the concurrent streaming management. The layers supported by WikiSensing are the *Sensor Layer* and the *Elastic Management* Layer. The *Sensor Layer* represents the storage of the data for the sensors and the *Elastic Management Layer* represents an elastic resource provision infrastructure

for the whole system. The other layers are the *Data Analysis Layer* that is used for information compression and the *Application Layer* used for application integration. The *Sensor Layer* is simulated in WikiSensing by generating over 100 sensor node records with specified location IDs and a sequence of readings. The database is maintained on the *IC Cloud* [85] computing infrastructure. WikiSensing *API* provides the capability for each node for receiving quires and sending response. The *Elastic Management Layer* is supported by integrating the *EIMAP* scheduling algorithm into the *Optimization* module of the *Data Aggregation* component of WikiSensing (Figure 3.1).

The Application Layer is simulated by the *Siege* benchmarking that mimics the users' accessing a web server with a configurable number of concurrent simulated users. The performance of *EIMAP* is measured using this benchmark by identifying how it stands up to load on the internet based on the duration of the transactions and the number of simulated users. The reaction from *Ma et al*. on the performance *EIMAP* system was that the design of the algorithm and the data management (based on WikiSensing) provided higher performance in energy efficiency and system response speed.

9.2.4. A Cloud-based Sensor Informatics Platform

The goal of the research work by [133] is to utilise WikiSensing, a cloud based system as an informatics platform for sensing applications for digital life. The challenges were based on high costs of redundant data measurements, excessive and inflexible resource utilization for processing, managing, and storing data as well as the difficulty in extracting information from the observed data of sensor systems.

WikiSensing provides the sensor data acquisition and is the management layer of this informatics platform. The focus of WikiSensing is on sensor informatics, instead of physical sensors or sensor network protocol design. The subsystem for sensor data acquisition provides the interface for sensor devices and

applications to transfer data to WikiSensing. In WikiSensing, collective data sampling is designed to support the sensor data collection in a wiki-style way: for a single event, multiple sensor data from different sources and other approaches (like social media) could be submitted to collaboratively describe the sensory target. In the case when sensor data are not adequate, or when a sensor measurement at a specific location is unavailable, the virtual sensing features of WikiSensing is adopted, which applies spatial interpolation among existing measurements to compensate for the insufficient or missing data. In addition to raw sensor data, the WikiSensing ontology of the sensors that describes their technical characteristics is also stored in an ontology repository. These raw data and ontology data support various data services, including querying and streaming.

9.3. Conclusion and Future Work

A wide area of work on the domain of collaborative sensor data management and trustworthiness assessment has been investigated in this research. As a result new challenges have emerged due to this research work. Hence several ideas and models can be established by further investigating these new challenges in greater depth. The following section presents the conclusions of this thesis and summarises the future work by identifying areas with potential for further research.

9.3.1. Interoperability for Sharing Data and Improve Performance

It is clear that the convergence of online collaborations with sensor data management can enable better use and understanding of the vast amounts of sensor information. Furthermore the efforts required are considerably lower due to the collaborative nature and the involvement of users with experience and knowledge. However due to large volumes of collaborative data from various sources leads to decline or weakening of interoperability among this data. Interoperability is required for effective sharing as well as for data analysis.

The intension is to develop a *Wiki Analytical* layer for the sensor and wiki data that can mark-up the information using a universal methodology. The objective of introducing a *Wiki Analytical* layer is to use the gathered sensor data and put it into further analysis so that it can provide useful insights. For example, it is useful to know whether there are relationships between the temperature and the *ozone* pollution level of an environment or links between the noise levels and prevailing traffic. Hence the data in the system must be transformed into a suitable format in order to be further analysed. It is possible to make such transformation by adding a new layer to the existing WikiSensing architecture.

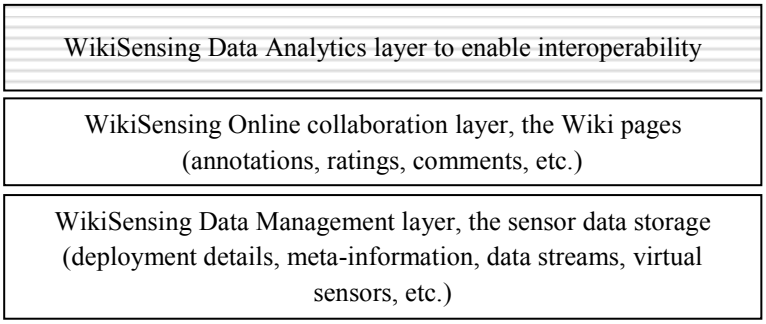


Figure 9.2: The proposed new layer for standardising sensor and Wiki data

The proposed new layer (highlighted in Figure 9.2) would enable the existing data and information to be formatted and annotated based on standard mark-up. This tier would also be able to extract and use the information from the Wiki pages created as a result of online collaborations. Hence the goal is to provide a platform to annotate this information so that it can be further analysed. This will clearly increase the chances of obtaining useful insights using the rich set of underlining sensor and Wiki data.

The aggregate operations discussed in this thesis can involve a large amount of sensors and sensor measurements. Moreover the data involved in aggregations can be extremely high as in cases where virtual sensors are created using other virtual sensors. Hence there is the need to concentrate on enhancing the response time of these extensively used aggregate queries.

The response time of aggregate operations is expected to be reduced by using the *MapReduce* in *MongoDB* [134] for batch processing of data. This is similar to Apache *Hadoop* (*hadoop.apache.org*) but uses distributed processing of large data sets across clusters of computers. *MapReduce* in *MongoDB* processes the input from a collection and outputs it to a collection. This can be used for the aggregation queries especially when they involve combining a large number of sensor data streams. This relates to the work of [135] that proposes a scalable platform for network log analysis, which targets for fast aggregation and agile querying.

9.3.2. Effective usage of Contextual data, Improve Estimation of Views of Experts, Incorporating Reputation Management and Trust Assessment for other Collaborative data domains

Approximating user input can be a challenging task as experienced when estimating the views of experts (chapter 7). It was clear that in some cases when there was insufficient data estimating user ratings can end up mutating the training data and resulting in a lower accuracy. Hence a future direction of research would be to identify techniques that can more accurately approximate user input or ratings. One potential solution is to use a combination of the existing views of expert estimation strategies. Another approach is to use the comments and annotations provided on the Wiki as an additional source of data for modelling user ratings.

When assessing trustworthiness for environmental sensors, contextual data helped identify certain conditional dependencies between metrics when training models. However using contextual data to assess trustworthiness for route traces require further investigation as most of them (e.g. lifts, staircases, etc.) do not directly affect the sensor measurement. Moreover contextual data in this domain usually impacts the trustworthiness subjectively, based on perspective and ability of visually handicapped people. Hence the challenge will be to understand how these contextual data can be used with other trust metrics. It will also be interesting

to identify if this data can be directly used in training or if certain transformations are required (e.g. from non-numerical to a range of numerical values).

The trustworthiness management framework provided trust ratings for sensors or sensor measurements. Hence it will also be important to monitor how the reputation of these trustworthiness ratings changes with time. The work by *Yu et al*. [136] provides useful foundations for reputation management and trust evolution that is based on the interaction of agents in multiagent environments. Their research has identified that explicit reputation management can help the agents detect selfish, antisocial, or unreliable agents. This can be incorporated with WikiSensing trust framework to understand behaviours of sensors based on their changes in reputation. Moreover the advantage when adopting a reputation management framework in WikiSensing when compared to the work by *Yu et al.* is that the data is available centrally without the need to exchange information as in the case of agents. Hence one approach will be to initially assume the sensors to be trustworthy and then monitor how their reputation changes. The reputation of a sensor can effectively be the difference between trust ratings at different time frames. For example, a decline in the reputation can be seen when the sensor gets a lower trust rating. Moreover changes of the reputation will be useful to identify certain trends and pattern of sensors. For example, it will be possible to see if and when untrustworthy sensors later become trustworthy or if they tend to continuously stay untrustworthy. This information can also be used in virtual sensors that combine data streams. For instance, if the reputation of a sensor keeps diminishing, it will be a good indication to remove or discount such stream from the virtual sensor.

In this research the trust models were used to determine the trustworthiness specifically in the area of sensor data. Clearly an innovative approach will be to see how these models can be applied or be extended to manage the trustworthiness of other collaborative data systems (e.g. *Facebook*, *Wikipedia*, *Twitter*, etc.). The plan is to calculate a set of metrics based on current and historical data, train the models and determine trust. Naturally the challenge will be to recognise and calculate a set

of metrics that can correctly reflect on the trustworthiness as well as to identify suitable methods to represent them to determine trustworthiness. Work is currently carried out in capturing data for metrics in *Twitter* to assess trustworthiness of feeds. Some of these metrics are based on the number of tweets, re-tweets, followers and contextual data on locations, information on timelines, categorised or hash tagged key words, etc.

9.3.3. Concluding Remarks

The architectural design and implementation of WikiSensing provided a framework for online collaborative sensor data management. It addressed the challenges of managing large volumes of real-time, heterogeneous data and demonstrated how collaborative information can be organised and represented. It also explained the use of virtual sensors and how they can efficiently query data by modelling the overlapping of information.

The trustworthiness management framework successfully captured, calculated and represented trust metrics for sensor data. The trust models based on Bayesian probabilistic modelling proved to be effective in determining trustworthiness of sensor data. The Bayesian Network model provided the most accurate results as it took into account conditional dependencies among metrics. Moreover the Naïve Bayesian models were simpler to implement. In addition the usage of continuous data showed better results as opposed to binary data. Further employing smaller calculation windows was suitable for the early detection of untrustworthy measurements. It was also identified that incorporating supplementary information such as the views of experts metric resulted in better accuracy when determining the trustworthiness of sensor data. Furthermore the trust models were also successfully used to determine the trustworthiness of route traces that were represented in a multilevel of information. It was also understood that in a multilevel of information, metric were needed to identify correlations of data. In addition Bayesian modelling proved to be successful in composing trust values as opposed to averaging or voting.

Bibliography

1. Wang, K.-C. and P. Ramanathan, *Collaborative sensing using sensors of uncoordinated mobility*, in *Distributed Computing in Sensor Systems*. 2005, Springer. p. 293-306.

2. Lindsey, S. and C.S. Raghavendra. *PEGASIS: Power-efficient gathering in sensor information systems*. in *Aerospace conference proceedings, 2002. IEEE*. 2002. IEEE.

3. Balazinska, M., et al., *Data management in the worldwide sensor web.* Pervasive Computing, IEEE, 2007. 6(2): p. 30-40.

4. Bafoutsou, G. and G. Mentzas, *Review and functional classification of collaborative systems.* International journal of information management, 2002. 22(4): p. 281-305.

5. Manyika, J., et al., *Big data: The next frontier for innovation, competition, and productivity.* 2011.

6. Pottie, G.J. and W.J. Kaiser, *Wireless integrated network sensors.* Communications of the ACM, 2000. 43(5): p. 51-58.

7. Murty, R.N., et al. *Citysense: An urban-scale wireless sensor network and testbed*. in *Technologies for Homeland Security, 2008 IEEE Conference on*. 2008. IEEE.

8. Silva, D., M. Ghanem, and Y. Guo, *WikiSensing: An Online Collaborative Approach for Sensor Data Management.* Sensors, 2012. 12(10): p. 13295-13332.

9. Folea, S., M. Ghercioiu, and D. Ursutiu, *Cloud instrument powered by solar cell sends data to pachube.* International Journal of Online Engineering (iJOE), 2010. 6(4): p. pp. 20-25.

10. Nikzad, N., et al., *CitiSense: Adaptive Services for Community-driven Behavioral and Environmental Monitoring to Induce Change*. 2011: Department of Computer Science and Engineering, University of California, San Diego.

11. Grosky, W.I., et al., *Senseweb: An infrastructure for shared sensing.* Multimedia, IEEE, 2007. 14(4): p. 8-13.

12. Hijikihigawa, M. and S. Kataoka, *Sensor device*. 1992, Google Patents.

13. Marsoner, H., et al., *Sensor device*. 1992, Google Patents.

14. Ganesan, D., D. Estrin, and J. Heidemann, *DIMENSIONS: Why do we need a new data handling architecture for sensor networks?* ACM SIGCOMM Computer Communication Review, 2003. 33(1): p. 143-148.

15. Akyildiz, I.F., et al., *A survey on sensor networks.* Communications magazine, IEEE, 2002. 40(8): p. 102-114.

16. Yick, J., B. Mukherjee, and D. Ghosal, *Wireless sensor network survey.* Computer networks, 2008. 52(12): p. 2292-2330.

17. Madden, S.R., et al., *TinyDB: An acquisitional query processing system for sensor networks.* ACM Transactions on Database Systems (TODS), 2005. 30(1): p. 122-173.

18. Madden, S. and M.J. Franklin. *Fjording the stream: An architecture for queries over streaming sensor data*. in *Data Engineering, 2002. Proceedings. 18th International Conference on*. 2002. IEEE.

19. Haklay, M. and P. Weber, *Openstreetmap: User-generated street maps.* Pervasive Computing, IEEE, 2008. 7(4): p. 12-18.

20. Abadi, D.J., et al., *Aurora: a new model and architecture for data stream management.* The VLDB Journal—The International Journal on Very Large Data Bases, 2003. 12(2): p. 120-139.

21. Bonnet, P., J. Gehrke, and P. Seshadri. *Towards sensor database systems*. in *Mobile Data Management*. 2001. Springer.

22. Richards, M., et al., *Grid-based analysis of air pollution data.* Ecological modelling, 2006. 194(1): p. 274-286.

23. Lynch, C., *Big data: How do your data grow?* Nature, 2008. 455(7209): p. 28-29.

24. Yao, Y. and J. Gehrke, *The cougar approach to in-network query processing in sensor networks.* ACM Sigmod Record, 2002. 31(3): p. 9-18.

25. Chang, F., et al., *Bigtable: A distributed storage system for structured data.* ACM Transactions on Computer Systems (TOCS), 2008. 26(2): p. 4.

26. Cheng, R. and S. Prabhakar, *Managing uncertainty in sensor database.* ACM SIGMOD Record, 2003. 32(4): p. 41-46.

27. Lu, C., et al. *Rap: A real-time communication architecture for large-scale wireless sensor networks*. in *Real-Time and Embedded Technology and Applications Symposium, 2002. Proceedings. Eighth IEEE*. 2002. IEEE.

28. He, T., et al. *SPEED: A stateless protocol for real-time communication in sensor networks*. in *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*. 2003. IEEE.

29. *Esper*. Last Access Data: 2014 May [cited 2014 May]; Available from: esper.codehaus.org.

30. *SQLStream*. Last Access Date: 2014 May [cited 2014 May]; Available from: www.sqlstream.com.

31. Gurgen, L., et al. *SStreaMWare: a service oriented middleware for heterogeneous sensor data management*. in *Proceedings of the 5th international conference on Pervasive services*. 2008. ACM.

32. Chamberlin, D., J. Robie, and D. Florescu, *Quilt: An XML query language for heterogeneous data sources*, in *The World Wide Web and Databases*. 2001, Springer. p. 1-25.

33. Milo, T. and S. Zohar. *Using schema matching to simplify heterogeneous data translation*. in *VLDB*. 1998. Citeseer.

34. Bray, T., et al., *Extensible markup language (XML)*. World Wide Web Journal, 1997. 2(4): p. 27-66.

35. Maier, D., *The theory of relational databases*. Vol. 11. 1983: Computer science press Rockville.

36. Membrey, P., E. Plugge, and T. Hawkins, *The definitive guide to MongoDB: the noSQL database for cloud and desktop computing*. 2010: Apress.

37. *MongoDb GridFS*. Last Access Date: 2014 March [cited 2012 August]; Available from: docs.mongodb.org/manual/core/gridfs.

38. Aberer, K., M. Hauswirth, and A. Salehi, *The Global Sensor Networks middleware for efficient and flexible deployment and interconnection of sensor networks.* Ecole Polytechnique Fdrale de Lausanne (EPFL), Tech. Rep. LSIR-REPORT-2006-006, 2006.

39. Bertino, E. and H.-S. Lim, *Assuring Data Trustworthiness-Concepts and Research Challenges*, in *Secure Data Management*. 2010, Springer. p. 1-12.

40. Dai, C., et al., *An approach to evaluate data trustworthiness based on data provenance*, in *Secure Data Management*. 2008, Springer. p. 82-98.

41. Rajagopalan, R. and P.K. Varshney, *Data aggregation techniques in sensor networks: A survey.* 2006.

42. Fox, G., et al. *A collaborative sensor grids framework*. in *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*. 2008. IEEE.

43. Donnellan, A., et al. *QuakeSim: Efficient Modeling of Sensor Web Data in a Web Services Environment*. in *Aerospace Conference, 2008 IEEE*. 2008. IEEE.

44. Jayasumana, A.P., Q. Han, and T.H. Illangasekare. *Virtual sensor networks-A resource efficient approach for concurrent applications*. in *Information Technology, 2007. ITNG'07. Fourth International Conference on*. 2007. IEEE.

45. Leuf, B. and W. Cunningham, *The Wiki way: quick collaboration on the Web.* 2001.

46. Adamic, L.A., et al. *Knowledge sharing and yahoo answers: everyone knows something*. in *Proceedings of the 17th international conference on World Wide Web*. 2008. ACM.

47. Resnick, P. and H.R. Varian, *Recommender systems.* Communications of the ACM, 1997. 40(3): p. 56-58.

48. Breese, J.S., D. Heckerman, and C. Kadie. *Empirical analysis of predictive algorithms for collaborative filtering*. in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. 1998. Morgan Kaufmann Publishers Inc.

49. Gowers, T. and M. Nielsen, *Massively collaborative mathematics.* Nature, 2009. 461(7266): p. 879-881.

50. *Mathematical Related Discussions*. Last Access Date: 2013 December [cited 2011 August]; Available from: ww.gowers.wordpress.com/2009/02/01/questions-of-procedure.

51. Grandison, T. and M. Sloman, *A survey of trust in internet applications.* Communications Surveys & Tutorials, IEEE, 2000. 3(4): p. 2-16.

52. Artz, D. and Y. Gil, *A survey of trust in computer science and the semantic web.* Web Semantics: Science, Services and Agents on the World Wide Web, 2007. 5(2): p. 58-71.

53. Bonatti, P., et al., *An integration of reputation-based and policy-based trust management.* networks, 2007. 2(14): p. 10.

54. Ganeriwal, S., L.K. Balzano, and M.B. Srivastava, *Reputation-based framework for high integrity sensor networks.* ACM Transactions on Sensor Networks (TOSN), 2008. 4(3): p. 15.

55. Blaze, M. and A.D. Keromytis, *The KeyNote trust-management system version 2.* 1999.

56. Blaze, M., J. Feigenbaum, and J. Lacy. *Decentralized trust management*. in *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*. 1996. IEEE.

57. Neisse, R., M. Wegdam, and M. Van Sinderen. *Trustworthiness and quality of context information*. in *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for*. 2008. IEEE.

58. Gomez, L., A. Laube, and A. Sorniotti. *Trustworthiness assessment of wireless sensor data for business applications*. in *Advanced Information Networking and Applications, 2009. AINA'09. International Conference on*. 2009. IEEE.

59. Gomez, L., X. Gentile, and M. Riveill. *A framework for trust assessment of sensor data*. in *Wireless and Mobile Networking Conference (WMNC), 2011 4th Joint IFIP*. 2011. IEEE.

60. Jøsang, A., *A logic for uncertain probabilities.* International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 2001. 9(03): p. 279-311.

61. Heckerman, D., D. Geiger, and D.M. Chickering, *Learning Bayesian networks: The combination of knowledge and statistical data.* Machine learning, 1995. 20(3): p. 197-243.

62. Mengshoel, O.J., A. Darwiche, and S. Uckun. *Sensor validation using Bayesian networks*. in *Proc. 9th International Symposium on Artificial Intelligence, Robotics, and Automation in Space (iSAIRAS-08)*. 2008.

63. Sun, Y.L., et al., *Information theoretic framework of trust modeling and evaluation for ad hoc networks.* Selected Areas in Communications, IEEE Journal on, 2006. 24(2): p. 305-317.

64. Aloisio, G., et al. *Globus monitoring and discovery service and SensorML for grid sensor networks*. in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE'06. 15th IEEE International Workshops on*. 2006. IEEE.

65. Niles, I. and A. Pease. *Origins of the IEEE standard upper ontology*. in *In Working Notes of the IJCAI-2001 Workshop on the IEEE Standard Upper Ontology*. 2001. Citeseer.

66. Russomanno, D.J., C.R. Kothari, and O.A. Thomas. *Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models*. in *IC-AI*. 2005.

67. McGuinness, D.L. and F. Van Harmelen, *OWL web ontology language overview*. W3C recommendation, 2004. 10(2004-03): p. 10.

68. Eid, M., R. Liscano, and A. El Saddik. *A universal ontology for sensor networks data*. in *Computational Intelligence for Measurement Systems and Applications, 2007. CIMSA 2007. IEEE International Conference on*. 2007. IEEE.

69. Chidamber, S.R. and C.F. Kemerer, *A metrics suite for object oriented design.* Software Engineering, IEEE Transactions on, 1994. 20(6): p. 476-493.

70. Sabater, J. and C. Sierra, *Review on computational trust and reputation models.* Artificial Intelligence Review, 2005. 24(1): p. 33-60.

71. Krukow, K., M. Nielsen, and V. Sassone, *Probabilistic Computational Trust.* 2009.

72. Despotovic, Z. and K. Aberer, *P2P reputation management: Probabilistic estimation vs. social networks.* Computer Networks, 2006. 50(4): p. 485-500.

73. Kuter, U. and J. Golbeck. *Sunny: A new algorithm for trust inference in social networks using probabilistic confidence models*. in *AAAI*. 2007.

74. Nardi, B.A., *Activity theory and human-computer interaction.* Context and consciousness: Activity theory and human-computer interaction, 1996: p. 7-16.

75. Liberty, J., *Programming C#: Building. NET Applications with C#*. 2009: O'reilly.

76. *12 Advantages of ASP.NET*. Last Access Date: 2014 May [cited 2014 May]; Available from: blog.seekdotnet.com/asp-net/12-advantages-of-asp-net.

77. *MediaWiki*. Last Access Data: 2014 May [cited 2014 May]; Available from: www.mediawiki.org/wiki/Manual:Deciding_which_wiki_software_to_use.

78. O'reilly, T., *What is web 2.0*. 2005.

79. Svennerberg, G., *Beginning Google Maps API 3*. 2010: Apress.

80. Veit, M. and S. Herrmann. *Model-view-controller and object teams: A perfect match of paradigms*. in *Proceedings of the 2nd international conference on Aspect-oriented software development*. 2003. ACM.

81. *MVC Framework*. Last Access Date: March 2014 [cited 2011 October]; Available from: weblogs.asp.net/scottgu/archive/2007/10/14/asp-net-mvc-framework.aspx.

82. DuBois, P., *MySQL*. 2008: Pearson Education.

83. Chodorow, K., *MongoDB: the definitive guide*. 2013: O'Reilly.

84. Harris, S., N. Lamb, and N. Shadbolt. *4store: The design and implementation of a clustered RDF store*. in *5th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2009)*. 2009.

85. Guo, L., Y. Guo, and X. Tian. *IC cloud: a design space for composable cloud computing*. in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. 2010. IEEE.

86. Cruz, I., H. Xiao, and F. Hsu. *An ontology-based framework for XML semantic integration*. in *Database Engineering and Applications Symposium, 2004. IDEAS'04. Proceedings. International*. 2004. IEEE.

87. *C# .Net Library for RDF*. Last Access Date: 2014 January [cited 2012 July]; Available from: ww.dotnetrdf.org.

88. Varley, I.T., et al., *No relation: The mixed blessings of non-relational databases.* 2009.

89. Dieberger, A., et al., *Social navigation: techniques for building more usable systems.* interactions, 2000. 7(6): p. 36-45.

90. Shumaker, B. and R. Sinnott, *Astronomical computing: 1. Computing under the open sky. 2. Virtues of the haversine.* Sky and telescope, 1984. 68: p. 158-159.

91. Crosbie, T.M.M. and C.W. Knouse, *Query string processing*. 2012, Google Patents.

92. Josefsson, S., *The base16, base32, and base64 data encodings.* 2006.

93. Curbera, F., et al., *Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI.* Internet Computing, IEEE, 2002. 6(2): p. 86-93.

94. Richardson, L. and S. Ruby, *RESTful web services*. 2008: O'Reilly.

95. Pautasso, C., O. Zimmermann, and F. Leymann. *Restful web services vs. big'web services: making the right architectural decision*. in *Proceedings of the 17th international conference on World Wide Web*. 2008. ACM.

96. Lopez, I.F.V., R.T. Snodgrass, and B. Moon, *Spatiotemporal aggregate computation: A survey.* Knowledge and Data Engineering, IEEE Transactions on, 2005. 17(2): p. 271-286.

97. Rabinovich, M. and O. Spatscheck, *Web caching and replication.* Sigmod Record, 2003. 32(4): p. 107.

98. *The Siege Benchmark*. Last Access Date: 2014 March [cited 2012 June]; Available from: ww.joedog.org/siege-manual. .

99. Subbarayan, S. and D.K. Pradhan. *NiVER: Non-increasing variable elimination resolution for preprocessing SAT instances*. in *Theory and Applications of Satisfiability Testing*. 2005. Springer.

100. Nah, F.F.-H., *A study on tolerable waiting time: how long are web users willing to wait?* Behaviour & Information Technology, 2004. 23(3): p. 153-163.

101. Burdick, D., M. Calimlim, and J. Gehrke. *MAFIA: A maximal frequent itemset algorithm for transactional databases*. in *Data Engineering, 2001. Proceedings. 17th International Conference on*. 2001. IEEE.

102. Ledlie, J., C. Ng, and D.A. Holland. *Provenance-aware sensor data storage*. in *Data Engineering Workshops, 2005. 21st International Conference on*. 2005. IEEE.

103. Crocker, D. and P. Overell, *Augmented BNF for syntax specifications: ABNF.* 2008.

104. Han, J., M. Kamber, and J. Pei, *Data mining: concepts and techniques*. 2006: Morgan kaufmann.

105. VIM, I., *ISO/TC 213 N 658.* International Organization, 2004. 2004: p. 09-14.

106. Nielsen, M., K. Krukow, and V. Sassone, *A bayesian model for event-based trust.* Electronic Notes in Theoretical Computer Science, 2007. 172: p. 499-521.

107. Pérez, J., M. Arenas, and C. Gutierrez, *Semantics and Complexity of SPARQL*, in *The Semantic Web-ISWC 2006*. 2006, Springer. p. 30-43.

108. Microsoft. *Naive Bayes Classification with C#.* Last Access Date: 2014 March [cited 2012 December]; Available from: msdn.microsoft.com/en-us/magazine/jj891056.aspx.

109. Fenton, N. and M. Neil, *Combining evidence in risk analysis using bayesian networks.* Agena White Paper W, 2004. 704.

110. Wang, Y. and M.P. Singh, *Evidence-based trust: A mathematical model geared for multiagent systems.* ACM Transactions on Autonomous and Adaptive Systems (TAAS), 2010. 5(4): p. 14.

111. Zadeh, L.A., *A simple view of the Dempster-Shafer theory of evidence and its implication for the rule of combination.* AI magazine, 1986. 7(2): p. 85.

112. Jøsang, A., R. Hayward, and S. Pope. *Trust network analysis with subjective logic*. in *Proceedings of the 29th Australasian Computer Science Conference-Volume 48*. 2006. Australian Computer Society, Inc.

113. Cano, A., A.R. Masegosa, and S. Moral, *A method for integrating expert knowledge when learning Bayesian networks from data.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2011. 41(5): p. 1382-1394.

114. Calder, B., et al. *Windows Azure Storage: a highly available cloud storage service with strong consistency*. in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. 2011. ACM.

115. Terveen, L. and W. Hill, *Beyond recommender systems: Helping people help each other.* HCI in the New Millennium, 2001. 1: p. 487-509.

116. Resnick, P., et al. *GroupLens: an open architecture for collaborative filtering of netnews*. in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. 1994. ACM.

117. Bilenko, M., et al., *Adaptive name matching in information integration.* Intelligent Systems, IEEE, 2003. 18(5): p. 16-23.

118. Harper, F.M., et al., *An economic model of user rating in an online recommender system*, in *User Modeling 2005*. 2005, Springer. p. 307-316.

119. Marlin, B.M. *Modeling user rating profiles for collaborative filtering*. in *Advances in neural information processing systems*. 2003.

120. Cheung, K.-W. and L.F. Tian, *Learning user similarity and rating style for collaborative recommendation.* Information Retrieval, 2004. 7(3-4): p. 395-410.

121. *Smart Cities, Ranking of European medium-sized cities*. Last Access Date: 2013 December [cited 2013 July]; Available from: ww.smart-cities.eu.

122. Hernández-Muñoz, J.M., et al., *Smart cities at the forefront of the future internet*, in *The future internet*. 2011, Springer. p. 447-462.

123. Lacey, G. and K. Dawson-Howe. *Evaluation of robot mobility aid for the elderly blind*. in *Proceedings of the Fifth International Symposium on Intelligent Robotic Systems*. 1997. Citeseer.

124. Lacey, G. and K.M. Dawson-Howe, *The application of robotics to a mobility aid for the elderly blind.* Robotics and Autonomous Systems, 1998. 23(4): p. 245-252.

125. Cohen, J., *Statistical power analysis for the behavioral sciencies*. 1988: Routledge.

126. Petrie, H., et al., *MoBIC: Designing a travel aid for blind and elderly people.* Journal of Navigation, 1996. 49(01): p. 45-52.

127. Hunt, R., *Percent agreement, Pearson's correlation, and kappa as measures of inter-examiner reliability.* Journal of Dental Research, 1986. 65(2): p. 128-130.

128. Birch, D., et al., *Paper: Concinnity: A Digital City Exchange Platform.*

129. Lee, C.-H., et al. *Building a generic platform for big sensor data application*. in *Big Data, 2013 IEEE International Conference on*. 2013. IEEE.

130. van Dam, K.H., et al. *Introducing a model composition platform for urban energy and transport systems*. in *In Proceedings of the Agent Technologies in Energy Systems workshop (ATES2012) at AAMAS2012*. 5 June 2012.

131. Huettel, S.A., A.W. Song, and G. McCarthy, *Functional magnetic resonance imaging*. Vol. 1. 2004: Sinauer Associates Sunderland.

132. Ma, Y., et al., *Elastic information management for air pollution monitoring in large-scale M2M sensor networks.*

133. Guo, Y., et al., *Wikisensing: Towards a cloud-based sensor informatics platform for life in a digital city.* Digital Futures, Aberdeen, UK, 2012: p. 23-25.

134. *MapReduce MongoDB*. Last Access Date: 2014 January [cited 2012 March]; Available from: ww.mongodb.org/display/DOCS/MapReduce.

135. Wei, J., et al. *Analysis farm: A cloud-based scalable aggregation and query platform for network log analysis*. in *Cloud and Service Computing (CSC), 2011 International Conference on*. 2011. IEEE.

136. Yu, B. and M.P. Singh. *An evidential model of distributed reputation management*. in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*. 2002. ACM.

Appendix

The appendix contains auxiliary information relating to WikiSensing's collaborative sensor data management and trustworthiness management framework that are referred from the main body of this thesis.

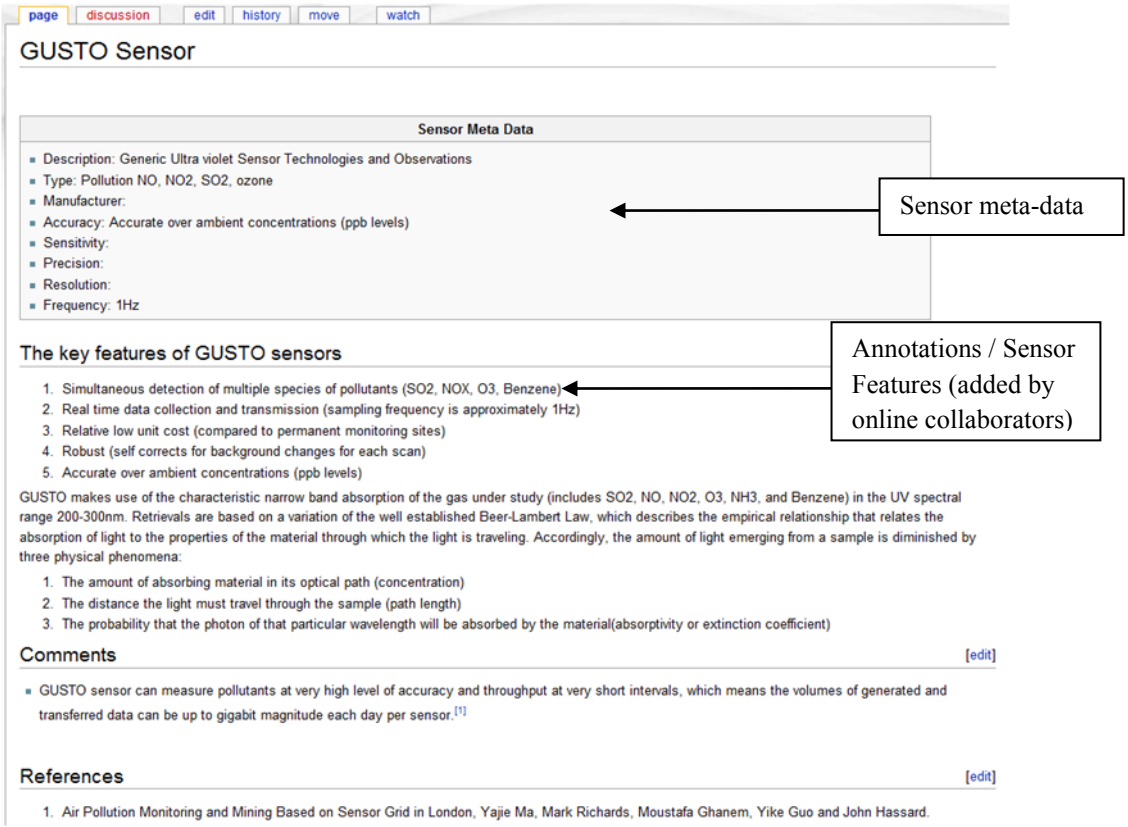


Figure A.1: Wiki pages to record annotations on sensor meta-data

1. Experimental results

The following confusion matrix illustrates the experimental results for the different views of experts metric (V) estimation methods discussed in chapter 6.

1. 100% sensor annotated

1.1. Available ratings extrapolated to entire sensor

		Predicted				
	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 200	FN 0	TP 182	FN 18	TP 199	FN 1
N	FP 12	TN 88	FP 0	TN 100	FP 11	TN 89

1.2. Available ratings extrapolated to entire sensor from annotated time frame

		Predicted				
	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 200	FN 0	TP 200	FN 0	TP 200	FN 0
N	FP 0	TN 100	FP 0	TN 100	FP 0	TN 100

1.3. Modelled based on similarities

		Predicted				
	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 200	FN 0	TP 198	FN 2	TP	FN 0
N	FP 11	TN 89	FP 11.5	TN 88.5	FP 11	TN 89

1.4. Unrated sensor measurement rated as 'Unknown'

		Predicted				
	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 200	FN 0	TP 193	FN 7	TP 200	FN 0
N	FP 12	TN 88	FP 12	TN 88	FP 5	TN 95

2. 75% sensor annotated

2. 1. Available ratings extrapolated to entire sensor

Predicted

	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 199	FN 1	TP 186	FN 14	TP 199	FN 1
N	FP 19	TN 81	FP 11	TN 89	FP 11	TN 89

2.2. Available ratings extrapolated to entire sensor from annotated time frame

Predicted

	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 200	FN 0	TP 200	FN 0	TP 200	FN 0
N	FP 10	TN 90	FP 11	TN 89	FP 3.5	TN 96.5

2.3. Modelled based on similarities

Predicted

	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 200	FN 0	TP 198	FN 2	TP 200	FN 0
N	FP 17	TN 83	FP 16	TN 84	FP 12	TN 88

2.4. Unrated sensor measurement rated as 'Unknown'

Predicted

	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 200	FN 0	TP 193	FN 7	TP 200	FN 0
N	FP 18	TN 82	FP 18	TN 82	FP 8	TN 92

3. 50% sensor annotated

3.1. Available ratings extrapolated to entire sensor

		Predicted				
	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 199	FN 1	TP 189	FN 11	TP	FN 1
N	FP 23	TN 77	FP 14.5	TN 85.5	FP 12	TN 88

3.2. Available ratings extrapolated to entire sensor from annotated time frame

		Predicted				
	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 200	FN 0	TP 198	FN 2	TP	FN 0
N	FP 17	TN 83	FP 14.5	TN 85.5	FP 6	TN 94

3.3. Modelled based on similarities

		Predicted				
	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 200	FN 0	TP 198	FN 2	TP 200	FN 0
N	FP 23	TN 77	FP 21	TN 79	FP 12	TN 88

3.4. Unrated sensor measurement rated as 'Unknown'

		Predicted				
	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 200	FN 0	TP 193	FN 7	TP 200	FN 0
N	FP 21	TN 79	FP 21	TN 79	FP 9	TN 91

4. 10% sensor annotated

4.1. Available ratings extrapolated to entire sensor

		Predicted				
	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 200	FN 0	TP 192	FN 8	TP 195	FN 5
N	FP 25	TN 75	FP 21	TN 79	FP 12	TN 88

4.2. Available ratings extrapolated to entire sensor from annotated time frame

		Predicted				
	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 200	FN 0	TP 198	FN 2	TP 200	FN 0
N	FP 22	TN 78	FP 21	TN 79	FP 10	TN 90

4.3. Modelled based on similarities

		Predicted				
	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 200	FN 0	TP 198	FN 2	TP 200	FN 0
N	FP 25	TN 75	FP 24	TN 76	FP 12	TN 88

4.4. Unrated sensor measurement rated as 'Unknown'

		Predicted				
	NB (Categorical)		NB (Continuous)		BN (Categorical)	
Actual	T	N	T	N	T	N
T	TP 200	FN 0	TP 193	FN 7	TP 200	FN 0
N	FP 24	TN 76	FP 24	TN 76	FP 12	TN 88

2. WikiSensing trustworthiness ontology

```
<owl:Class rdf:about="Trustworthiness">
   <owl:equivalentClass>
      <owl:Restriction>
         <owl:onProperty rdf:resource="determinedBy"/>
           <owl:onClass rdf:resource="Metrics"/>
            <owl:minQualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:minQualifiedCardinality>
            </owl:Restriction>
        </owl:equivalentClass>
    </owl:Class>
```

Figure A.2: A restriction imposed on the WikiSensing trustworthiness ontology

3. WikiSensing trustworthiness *API* services

Example requests and outputs of the available services for providing trustworthiness metrics that were used during the *UPLondon Crackathon* event.

1. Historical Information (*HI*)
   http:// wikisensing.org/WikiSensingTrustworthinessServiceAPI/A1/NO/200/2

```
<HistoricalInformation
xmlns="http://schemas.datacontract.org/2004/07/WikiSensingAPI"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
               <abnormalReadingPercentage>4</abnormalReadingPercentage>
               <lowerBound>2</lowerBound>
               <readingAverage>6.1</readingAverage>
               <sensitivity>2</sensitivity>
               <upperBound>10</upperBound>
               <window>200</window>
</HistoricalInformation>
```

Figure A.3: Sample output of WikiSensing trust services (*HI*)

2. Conflicts with Other sensors (*OS*)

   http://wikisensing.org/WikiSensingTrustworthinessServiceAPI/A1/NO/200/1/True

```
<ConflictSummary xmlns="http://schemas.datacontract.org/2004/07/WikiSensingAPI"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  <conflictDetails>
  <ConflictDetails>
  <neighbourSensorId>GUSTO_A1</neighbourSensorId>
  <reading>6.1</reading>
  <window>200</window>
  </ConflictDetails>

  <ConflictDetails>
  <conflict>0.02</conflict>
  <conflictExist>False</conflictExist>
 <conflictWithDistanceCoefficient>0.007</conflictWithDistanceCoefficient>
  <distance>111.19</distance>
  <distanceCoefficient>3</distanceCoefficient>
  <distanceType>Meters</distanceType>
  <neighbourSensorId>GUSTO_A2</neighbourSensorId>
  <reading>6.08</reading>
  <window>200</window>
  </ConflictDetails>

 <ConflictDetails>
  <conflict>31.92</conflict>
  <conflictExist>True</conflictExist>
  <conflictWithDistanceCoefficient>7.98</conflictWithDistanceCoefficient>
  <distance>156.17</distance>
  <distanceCoefficient>4</distanceCoefficient>
  <distanceType>Meters</distanceType>
  <neighbourSensorId>GUSTO_B2</neighbourSensorId>
  <reading>38.02</reading>
  <window>200</window>
  </ConflictDetails>

 <ConflictDetails>
  <conflict>28.04</conflict>
  <conflictExist>True</conflictExist>
  <conflictWithDistanceCoefficient>9.347</conflictWithDistanceCoefficient>
  <distance>109.66</distance>
  <distanceCoefficient>3</distanceCoefficient>
  <distanceType>Meters</distanceType>
  <neighbourSensorId>GUSTO_B1</neighbourSensorId>
  <reading>34.14</reading>
  <window>200</window>
  </ConflictDetails>
  </conflictDetails>
</ConflictSummary>
```

Figure A.4: Sample output of WikiSensing trust services (*OS*)