GEOMATICS
ENGINEERING

UCGE Reports
Number 20369

Department of Geomatics Engineering

The Automatic Grouping of Sensor Data Layers Using Semantic Clustering and Classification to Group Semantically Similar Sensor Data Layers

(URL: http://www.geomatics.ucalgary.ca/graduatetheses)

by

Ben Charles Knoechel

December 18, 2012

SCHULICH
School of Engineering
UNIVERSITY OF CALGARY

UNIVERSITY OF CALGARY

The Automatic Grouping of Sensor Data Layers

Using Semantic Clustering and Classification to Group Semantically Similar Sensor Data Layers

by

Ben Charles Knoechel

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF GEOMATICS ENGINEERING

CALGARY, ALBERTA

December, 2012

Abstract

The Sensor Web is a growing phenomenon where an increasing number of sensors are collecting data in the physical world, to be made available over the Internet. Open standards have been proposed and are being implemented to eliminate the problem of semantic interoperability, the goal being to allow systems to share data automatically. Spatial Data Infrastructures (SDIs) are tools that have been developed to manage geospatial data from many different sources. However, there are still problems with interoperability associated with a lack of standardized naming, even with data collected using the same open standard. The objective of this thesis is to automatically group similar sensor data layers. We propose a methodology to automatically group similar sensor data layers based on the phenomenon they measure. Our methodology is based on a unique bottom up approach that uses text processing, approximate string matching, and semantic string matching of data layers. Text processing includes normalization and tokenization to standardize syntactic differences in the naming. Approximate string matching techniques include Levenshtein Distance, a Length Adjusted Levenshtein Dissimilarity, Jaro Dissimilarity, JaroWinkler Dissimilarity, Jaccard Dissimilarity, and Cosine Dissimilarity. For semantic string matching, we use WordNet as a lexical database to compute word pair similarities and derive a set-based dissimilarity function using those similarity scores. These string matching algorithms are used to produce dissimilarity values between data layers, which are in turn used to provide data layer to data layer mappings, similar data layer clusters, and mapping between a set of class names and data layers. For clustering, we tested three different clustering algorithms, K-Medoids, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), and Hierarchical Agglomerative Clustering (HAC). We evaluate and discuss the results of our methodology, and introduce a proof of concept Virtual SOS service to show the utility of such research.

Acknowledgements

First, I would like to thank the University of Calgary. I also want to thank the faculty and staff of Geomatics engineering, who have helped support me since I started my undergraduate degree in 2005.

Thanks to Dr. Reda Alhajj for his informative and helpful lectures on Information Retrieval, which has shaped and guided my research.

I give thanks to Dr. Steve Liang for funding my research and giving me the opportunity to do my Masters degree. I also give thanks to the entire research group, the GeoSensorweb Lab, for helping me with my research, editing my thesis, and providing invaluable feedback, including Fanny Tsai, Mohammad Ali Jazayeri, and Leah Li. I owe Andrew Phillips thanks for reviewing my thesis, but more importantly for his moral support for the past two years through graduate school. I give special thanks to Alec Huang, who took it upon himself to help me find a good research topic, and guide me along the way. For the countless times he has spent reviewing my work and giving me advice, I will always be grateful.

I want to give a special thanks to my family. Their love and support have made it possible for me to excel in school and without them I would not be where I am today.

Finally, I wish to dedicate this thesis to the person I love and who has changed my life for the better in every way possible. To Candice Coley, with all the love I have to give.

Table of Contents

List of Tables

List of Figures and Illustrations

List of Algorithms

List of Symbols, Abbreviations and Nomenclature

Symbol	Definition
AI	Artificial Intelligence
DL	Description Logic
GIS	Geographic Information System
GML	Geographic Markup Language
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IR	Information Retrieval
LALD	Length Adjusted Levenshtein Distance
O&M	Observation and Measurements
OGC	Open Geospatial Consortium
OWL	Ontology Web Language
PL	Property Layer
PSM	Parametric String-based Matcher
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SDI	Spatial Data Infrastructure
SOS	Sensor Observation Service
SPARQL	The Simple Protocol and RDF Query Language
SWE	Sensor Web Enablement
tf-idf	Term Frequency-Inverse Document Frequency
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name

VMM	Vector-based Multi-word Matcher
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	eXtended Markup Language

Chapter 1

Introduction

The World Wide Web (WWW) has had a profound impact on almost all aspects of life. Over the past 20 years it came up from obscurity into the general public's consciences, and rightly so. The WWW has revolutionized communication. Although the Internet had been around for decades, the web was realized in the early 1990s. Tim Berners-Lee developed Hyper Text Markup Language (HTML), which allowed text documents to be shared via hyperlinks. As well, he developed protocols for sharing HTML, namely Hyper Text Transfer Protocol (HTTP). These technologies were the basis for the WWW, and with the availability of a user friendly browser, the WWW exploded in 1993. In a similar way, another looming technological paradigm is approaching, and its impact will be the next generation's WWW. This is known as the Sensor Web.

The term Sensor Web was first used by NASA [1], which was described as "the Sensor Web consists of a system of wireless, intra-communicating, spatially distributed sensor pods that can be easily deployed to monitor and explore new environments." [2] extended the definition of the Sensor Web to include a wide variety of applications and sensors. They discuss the wide variety of possible sensors, such as wireless sensor networks, flood gauges, weather towers, air pollution monitors, stress gauges on bridges, mobile bio-sensors, webcams, and satellite-borne earth imaging devices. As well, they argue that the Sensor Web can be thought of as a 'global sensor' that connects to all sensors and their observations. This extended definition of the Sensor Web is used throughout the rest of this thesis.

This Sensor Web will have an impact on all of our lives. "The next revolution of the Internet is not going to be built on manual input of information by 500 million or a billion users. Rather, there is much greater potential in connecting computers to sensors so that valuable new information can be created automatically without human data entry." [3]. The scale of such a Sensor Web is on the order of not millions, nor billions, but trillions. HP is covering the earth with over 1 trillion advanced sensor nodes and interconnecting them into an immense environmental network [4]. With the cost of embedded computers becoming cheaper, it is possible for people to start deploying their own sensors around their house.

To allow machines to communicate over the Sensor Web, a common language is needed. Just as the WWW has been successful due to the adaptation of HTML and HTTP, the Sensor Web will have a set of commonly used standards. The Open Geospatial Consortium (OGC), a standards organization, has been involved in developing these open standards for many years. They have developed the Sensor Web Enablement (SWE) standards. These standards define information models and communication protocols to enable various pieces of software on the Internet to share and interact with sensor data and sensors. Of particular importance is the Sensor Observation Service (SOS) standard. It is a service to communicate sensor data, and a core standard for sharing sensor data. In this thesis, the discussion of sensor data layers refers to data extracted from SOSs. For a more detailed examination of sensor data layers and the SOS standard, refer to chapter 3.

The nature of the Sensor Web is highly spatial-temporal. All physical sensors have some physical location, which makes all sensor data highly dependent on the proper modelling and understanding of location. As well, readings of events occur typically as instantaneous voltage readings, giving all sensor observations a timestamp. Observations and Measurements (O&M) is a OGC SWE standard for encoding sensor data [5], and the O&M model reinforces this spatial-temporal view of sensor data. The OM_Observation element has multiple time attributes such as resultTime, validTime, and phenomenonTime. The OM_Observation class does not explicitly contain location information. However, location information is provided by the feature of interest or by the observation procedure, according to the specific scenario [5].

In reaction to the huge amount of data to be generated from the Sensor Web, many research groups from all around the world are designing and researching systems to handle and process this new data. Geographic Information System (GIS) is the term commonly used to refer to software packages that are capable of integrating spatial and non-spatial data to yield the spatial information that is used for decision making [6]. The cooperation of multiple independent GISs is often referred to as a Spatial Data Infrastructure (SDI) [6]. [7] argue that the main components of a SDI include data providers, databases and metadata, data networks, technologies, institutional arrangements, policies and standards, and end-users. The concept of a SDI encompasses the technologies needed to harness the power of the Sensor Web.

There are many software components of the Sensor Web. One critical software component is a system which allows for the browsing of sensor data. This is important because the sheer volume of sensor data will make it difficult for end users to understand what data is available quickly. A general understanding of available data is necessary before data analysis can begin.

1.1 Problems

There are some problems that arise when SDIs try to connect to multiple data sources. Although the use of open standards eliminates many of the problems associated with interoperability, there are still real world access problems associated with data retrieval.

The rising number of data sources is making human interpretation impossible. Although a system can download and view all data according to the title, there may be too many titles, and too many different ways of encoding titles to make manual selection possible. For example, if a scientist is interested in collecting wind speed data, they have to manually parse many different titles of data, offered by the standard.

A major problem in the high number of unique data sources is the wide variety in the way data layers are named. Using open standards, the phenomenon the sensor is measuring can always be read in. However, the way phenomenon are named will vary, and basic string matching is insufficient to group them together.

For example, this research group has described the variety of assigned names to the observed properties in sensor data layers, using the SOS standard [8]. Table 1.1 shows the various observed properties, which all correspond to the same concept of wind speed. However, different data providers will label their data differently. This becomes a monumental issue when manually parsing thousands of unique layers.

Table 1.1: Various Observed Properties of the Concept Wind Speed

1	urn:x-ogc:def:property:OGC::WindSpeed
2	urn:ogc:def:property:universityofsaskatchewan:ip3:windspeed
3	urn:ogc:def:phenomenon:OGC:1.0.30:windspeed
4	urn:ogc:def:phenomenon:OGC:1.0.30:WindSpeeds
5	urn:ogc:def:phenomenon:OGC:windspeed
6	urn:ogc:def:property:geocens:geocensv01:windspeed
7	urn:ogc:def:property:noaa:ndbc:Wind Speed
8	urn:ogc:def:property:OGC::WindSpeed
9	urn:ogc:def:property:ucberkeley:odm:Wind Speed Avg MS
10	urn:ogc:def:property:ucberkeley:odm:Wind Speed Max MS
11	http://marinemetadata.org/cf#wind_speed
12	http://mmisw.org/ont/cf/parameter/winds

This problem becomes more difficult when trying to identify semantic relationships between data layers. "... relying on those URNs to perform string based search for sensor observables has serious drawbacks when it comes to realizing advanced sensor discovery tools as the meaning of the observables is ignored" [9]. One such example would be the relationship between 'precipitation' and 'rainfall'. Rainfall is a type of precipitation, and any search for precipitation should include all children concepts, such as 'rainfall', 'snowfall', 'hail', etc. Using a computer to define these relationships is very difficult, because although these concepts are intuitively related to any human, to any computer these are simply different sequences of characters.

1.2 Previous Solutions

One proposed method for finding and grouping similar sensor data layers is using a Sensor Observable Registry (SOR) and semantic annotations [9]. The SOR comprises of a dictionary of URNs identifying observed properties, as well as definitions of the observed properties and references to concepts for those observed properties in an ontology. This is a practical way to manage sensor data layers, but requires a certain level of manual work. New observed properties must be manually linked to some agreed upon ontology. As well, if multiple ontologies are used, then some method of matching different ontologies must be implemented. This solution fits into the work described in [10]. They describe a Sensor Plug and Play infrastructure, including a description on semantically-enabled matchmaking. Although they discuss the use of syntactic metrics for matching, their emphasis is on a large scale Sensor Web architecture. The focus is on grouping data layers from SOSs today, and with minimal help from the data provider. This is based on the assumption that many real world data providers will not be likely to register or annotate their sensors, and simply upload the data to SOS specifications.

A folksonomy-based recommendation system has been proposed to handle large volumes of sensor data [11]. Although these systems are very effective, these systems often suffer from cold-start problems. There is a potential benefit in building hybrid systems that utilize both external knowledge sources and user defined annotations, but that work is outside the scope of this thesis.

It is simply impossible to parse and scan what isn't available. A key assumption is made that all data providers are not willing to create or define an ontology, or that their naming convention does not follow a reference ontology. Instead of relying on data providers to provide semantic cues, the proposed methodology will consume text information directly from the open standards and use that data, along with some general-purpose lexical database, to infer semantics between data layers.

Another key assumption of the methodology is the ‘title’ of the sensor data layer is a concise word or phrase to describe the data. This assumption is justified because the use of open standards ensures consistency with the title, even if the data provider doesn’t use or know about a naming registry.

1.3 Challenges

Grouping similar data layers is not an easy task. There are two fundamental differences between similar labels, syntactic differences and semantic differences. Syntactic refers to various ways to encode and represent the same concept. For example, the three strings ‘windspeed’,‘WIND_SPEED’,‘ogc:urn:ucalgary:geomatics/Ontology.owl#Wind**Speed’ are all ways to encode the concept of wind speed. By using open standards many of the syntactic issues can be removed, such as character encoding. However, there is still the issue of whitespace, links, and other ways to represent the same concept.

There is also the notion of semantic differences, which is more subtle. A well known example is how one word ‘bank’ can represent a money lending institution or a part of a river. The more important problem is how to establish semantic relationships. Two concepts, ‘rain’ and ‘precipitation’ are very much related, although represented by two distinct concepts. The key to properly grouping layers is to be able to identify these relationships.

1.4 Contributions

This thesis contributes a great deal to the SDI community. The first and foremost contribution is the evaluation of various syntactic and semantic string functions for the purpose of grouping similar sensor data layers. This provides the community with some initial work on using a solid bottom up string matching approach for data interoperability. String processing and string matching can be more generally applied to other aspects of data interoperability in SDIs, and the evaluation shows how these techniques performed for the data set.

Another important contribution of this work is that it highlights the current progress of OGC's SWE, namely the SOS. This thesis discusses the SOS in detail, includes example data from currently deployed SOS data providers. This work focuses on the current problem in SOS regarding inconsistent naming, and serves as a record of the current progress of the OGC standard. This could possibly benefit those wishing to design and implement other open standards, both within and outside the GIS community.

A major contribution of this work is the unique data set used for both clustering and classification. As far as we know, there is no other research that has attempted to cluster or classify data layers. This is similar to some of the work done in clustering tags, except that this data set is fundamentally different from tags. This provides a unique case to those interested in Information Retrieval or data mining, on how techniques may vary from data set to data set.

I have researched, implemented, tested, and evaluated the methodology presented in this thesis. My research group, the GeoSensorweb Lab, has provided me a Java based SOS API for collecting data, and a list of SOS services to connect to. I have written software to process the SOS data, convert it into Property Layers (PLs), implemented dissimilarity functions between the PLs, and wrote the code to cluster and classify PLs into groups, as well as the code to evaluate it and produce figures. I used Java code for most of the work, with PostgreSQL database to store data, and R to generate figures. In Chapter 5, a Ph.D student wrote the Translation Engine part of VirtualSOS. The word pair similarity scores are generated using a Perl script from the original researchers. The clustering algorithms and dissimilarity functions were re-implemented in Java by me. Everything else in this thesis is work that I have written and researched.

The research in this thesis is currently being submitted as a journal paper to International

Journal of Geo-Information, “A Bottom-Up Approach for Automatically Grouping Sensor Data Layers by their Observed Property”. It currently undergoing revisions.

Chapter 2

Related Work

The process of grouping similar data layers in a SDI has not been described extensively in literature. The algorithms which our research group uses for grouping have been well defined in various other research fields. The Related Work chapter is divided as follows. First, we will look at a branch of computer science known as Information Retrieval (IR), because this field is well known and has been used to group together many other kinds of data, such as documents and web pages. This provides the framework for many of the other related works discussed.

Next, several research areas in which string matching algorithms are used to identify similar entities in data are discussed. This includes Record Linkage, which is the task of finding and linking records in a set of data. Schema matching is also a research area that utilizes string matching algorithms and structural components of data for integrating two heterogeneous data sets. Ontological alignment is another body of work which is also relevant for string matching and text processing..

Related works about clustering are discussed next. Clustering is used in the methodology to group similar data layers together. Both document and tag clustering are referenced, as they are the most related data type to sensor data layers.

Next, the notion of semantic similarity is introduced. WordNet is introduced, and followed by how WordNet can be related to semantic similarity.

Finally, it is important to understand ontologies, the Semantic Web, and Linked Data. These concepts are very relevant to task of inferring relationships between data. Although these approaches are not used in this thesis, a great deal of research in the GIS community involves the use of ontologies. Afterwards, ontologies are directly compared to the bottom-up methodology in this thesis.

2.1 Information Retrieval

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers) [12]. A well known example of IR is a search engine, where web pages are indexed and compared to user searches. The text of the web pages is compared to text of the query, and as such this string to string comparison is a useful body of work to study. In IR, documents consist of text documents, web pages, or other sources of text, hereon referred to as documents.

In IR, there are four major steps involved [12].

1. Collect documents to be indexed
2. Tokenize the text
3. Do linguistic preprocessing on tokens
4. Index the documents by building an inverted index

For example, lets say we are given all of Shakespeare's works, with each play as a document, and we want to perform some search on them. Linear searching on each document for every query is inefficient, so we need to construct an index structure, such as an inverted index. The documents are collected, and all the text is broken up into individual words, called tokens. All the tokens are counted for each document, as shown in 2.1. The *term frequency* is used in this index, which is simply the number of tokens that appear in each document. Using an inverted index, queries about the documents can easily be answered. For example, a search query may contain the term 'Juliet', and we can easily find books where the token Juliet appears more frequently.

There are some specific areas of IR that will be emphasized based on how they relate to our methodology.

Table 2.1: Example of an Inverted Index

Term/Document	Book 1	Book 2	Book 3
hello	4	5	0
light	2	8	0
Juliet	1	0	17

2.1.1 Tokenization and Normalization

In our methodology, it is necessary to compute some score to represent the relationship between two data layers. All the information about what the sensor data is measuring is stored as text data. All text based similarity or distance functions fall under two broad categories, *set based* or *edit based* [13]. Set based algorithms look at the number, position, and importance of common tokens. Edit based algorithms use strings of text as input. Before using these algorithms, the text data often needs to be processed, to make sure the computed score is truly meaningful. Text processing is discussed in the context of tokenization and normalization.

Tokenization is the process of converting text into distinct tokens, and normalization is the process of canonicalizing strings such that superficial differences between strings are removed. Together, they are used to break up very large chunks of text into distinct, normalized tokens.

The text data is simply a string, where a string is defined as a sequence of characters. Two strings are identical if and only if they have the same number of characters in the same order. Tokens are strings, except a token is meant to represent some distinct and meaningful piece of information. In IR, a token can be either a sentence, a word, a phrase, etc. For this work, a token is intended to refer to some English word.

Tokenization consists of breaking up long strings into tokens, such as splitting up some document into distinct words. A very simple approach would simply be to find delimiting characters, such as whitespace characters or periods, and use all the strings between them as tokens. Some problems with this approach are very obvious, such as how to deal with special characters, or with special patterns associated with some concepts. Compound nouns, for example, are nouns separated by whitespace, and those nouns by themselves do not mean the same thing. Table 2.2 shows some interesting examples for any tokenizer.

Table 2.2: Some Examples of Words with Distinct Patterns

Example Token	Meaning
C++	Programming Language
john.smith@ucalgary.ca	E-mail address
M*A*S*H	TV Show
http://ucalgary.ca	Website URL
m/sˆ2	Units
Mr. Smith	Last Name
San Francisco	City

Stop words are words that do not contribute to the meaning of the document, and often cause problems for IR. These are tokens such as 'a', 'or', 'and', 'the', and so forth. In building an index, a flat list of stop words is collected, and they are simply thrown away as potential token candidates.

Token normalization is performed to ensure that there are no redundant tokens in an index. For example, the two tokens 'usa' and 'U.S.A' may refer to the same concept. Alternatively, there may be value in making a distinction between two concepts such as 'cat', a house hold pet, and 'CAT', a company that constructs industrial machinery. Case-folding is a type of normalization, which takes all tokens and converts all uppercase characters into lowercase characters. Stemming is another method of normalization. Root words differ in context, ie. organize, organizes, organizing. The goal is to reduce inflectional forms and sometimes try to relate to some base form of the word.

2.1.2 tf-idf and the Vector Space Model

An important part of IR is how scoring works in IR such that queries are compared to an inverted index for the purpose of finding a notion of related documents given a query.

As noted earlier, the term frequency is a measure of the number of times a particular token appears in a document, $tf_{t,d}$. Here, tf refers to term frequency while t refers to the term, or token, and d refers to the document.

Some terms appear more frequently in text than other terms, and it is important to lower the impact of these very common words. The document frequency, df_t, is the number of times term t occurs in all documents. The inverse document frequency, idf_t, is

$$idf_t = \log \frac{N}{df_t} \tag{2.1}$$

Here, N refers to the total number of documents. The inverse document frequency limits the impact of terms that appears frequently across many documents. In other words, rare words have higher scores because they only appear in a few documents. Finally, the two measures can be combined to the commonly used tf-idf weighting, which is

$$tfidf_{t,d} = tf_{t,d} * idf_t \tag{2.2}$$

The tf-idf score is a commonly used value given to each term-document pair.

The Vector Space Model is intuitive. Here, we imagine that every document is a vector. Every vector is *M-dimensional*, where M is the number of words across all documents. The value of each dimension depends on the model that we use. For example, we could use word frequency to define these vectors. Using word frequency, from 2.1, we define the Book 1 vector as $(4, 2, 1)$, where the dimensions are the words hello, light, and Juliet, respectively. However, the Vector Space Model is commonly used with tf-idf weights. The advantage of the Vector Space Model is that it is very easy to compute the dot product between different documents for a similarity measure, or even between documents and queries, if a query is represented as a vector of words.

In our methodology, we use a simplified boolean model of a token either existing or not, because repeated words do not exist. However, the idea of treating data layers as vectors will be used for calculating similarity and dissimilarity scores.

2.2 Record Linkage

Record linkage is the means of combining information from a variety of computerized files [14]. Basic methods include comparing the name across pairs of files to determine those pairs of records that are associated with the same entity. Entities could be people, businesses, or some other type of unit. An important section of record linkage is using string comparators, because in many matching situations, it is not possible to compare two strings exactly because of typographical errors.

The paper [15] is a comparison of string distance metrics for name-matching tasks. The authors evaluated three categories of distance metrics: edit-distance like functions, token-based distance functions, and hybrid functions. Edit-distance functions include Levenshtein distance, Monger-Elkan distance function, Jaro metric and JaroWinkler metric. For token-based functions, they consider Jaccard similarity, cosine (tf-idf) similarity, Jensen-Shannon distance, as well as a method proposed by Fellegi and Sunter. Overall, they found the best performance from a hybrid scheme combining tf-idf weights with Jaro-Winkler string distance scheme.

In this methodology, the Levenshtein distance, Jaro metric, and JaroWinkler method are used. These metrics have been chosen so that several edit-distance metrics can be compared to one another.

2.3 Schema Matching

Schema matching is an interesting field of work. It involves creating a mapping between elements of two schemas, for various applications. The methodology of schema matching is similar to that of matching data layers. There are some linguistic approaches used in schema matching that is used in this methodology, described next.

The authors of [16] talk about various techniques for schema matching. The elements of two schemas are compared and then matches between elements are made, effectively producing a mapping between elements. They discuss using linguistic strategies, such as name matching. Name matching is accomplished in many different ways, such as equality of names, canonical name representation, equality of synonyms, equality of hypernyms, similarity of names, or user provided matches.

2.4 Ontological Alignment

Ontological alignment is the process of finding correspondences or mappings between semantically related entities. The system AgreementMaker has been developed by [17] to find these relationships. This system has a solid methodology that can be applied to the automatic grouping of sensor data layers. Classes from one ontology are compared to classes of another ontology, in order to find matching classes between the two. For the Parametric String-based Matcher (PSM), they take various parts of the ontology (localname, label, comments, etc), normalize them, apply some string metrics to develop similarity values, which are then weighted in a final similarity measure. The string matchers include edit-distance, JaroWinkler, and a substring-based measure devised by them. They also use a Vector-based Multi-word Matcher (VMM), which tokenizes ontological classes, builds tf-idf vectors and applies a cosine similarity.

This methodology is used as a general framework for the methodology in this thesis, except instead of ontological classes, sensor data layers and classes are used.

2.5 Clustering

Clustering analysis divides data into groups that are meaningful, useful, or both [18]. The clustering of data layers, to the best of our knowledge, is novel work. However, there are many other domains where clustering is used, include psychology, biology, statistics, pattern recognition, information retrieval, machine learning, and data mining [18]. We perform clustering to identify groups of related data layers, based on information extracted from the service provider. The focus of the clustering literature will be centered around document and tag clustering.

2.5.1 Document Clustering

Document clustering is a mature research field to study as it has been the subject of extensive research. Document clustering is often used in information retrieval and often includes indexing words in documents and using tf-idf weights for comparisons [12]. As well, literature from document clustering and text clustering is useful as it provides a solid methodological framework. For example, [19] provides a useful framework, including text preprocessing, concept-based analysis, concept-based document similarity, and clustering techniques. Elements of the text processing used in this paper is used in this methodology.

2.5.2 Tag Clustering

Tag clustering is a similar body of work in which tags are clustered together to form groups of tags. In this context, tags are keywords or terms associated to a piece of information, and they appear frequently in blogs and in social media. The authors of [20] use clustering to aggregate similar tags. They overcome syntactic and semantic variability in the tags. This is accomplished by using the normalized Levenshtein distance and cosine similarity based on tag co-occurrences [20].

The authors of [21] use tag clustering to reduce redundant or idiosyncratic tags. They use a modified hierarchical clustering algorithm based on cosine similarity between tags.

2.6 Semantic Similarity

An important aspect of this research is semantic similarity. The best example is that we may have two semantically similar types of data, ‘precipitation’ and ‘rainfall’. They are different words, but their meaning is very much related. Of course, it is important to find a way to establish a relationship between them. The idea of semantic interoperability was defined in [22] as the goal of interoperating GISs. Semantic interoperability is described as “... is to provide seamless communication between remote GISs without having prior knowledge of the underlying semantics.” They go on to note that semantic heterogeneity is when a real world fact may have more than one underlying description.

Kuhn mentions the idea of a semantic reference system [23], and describes semantic interoperability as the capacity of information systems or services to work together without the need for human intervention.

The idea semantic similarity is introduced to help test whether or not terms and descriptions from one data source to another are related. This measure would overcome the semantic interoperability issues associated with heterogeneous naming sensor data layers.

Semantic similarity is a measure of how related two concepts or words are, and we base semantic similarity on the work done in [24]. They use WordNet [12] to define semantic relationships between words and concepts. WordNet is a lexical network of English words, and several measures of semantic similarity can be used to define measures between concepts and words.

By utilizing semantic similarity naming issues can be overcome to better group together related sensor data layers. However, there has been a lot of research into ontologies as a solution to semantic interoperability in SDIs. Although we do not utilize ontologies in our methodology, we must describe what they are, how they have been used in the GIS community, and why our approach does not use them.

2.7 Ontologies and Knowledge Representation

As mentioned in Chapter 1, ontologies have not been integrated into open standards. As a result, ontologies are not available to scan and parse in the real world. However, there has been a great deal of research into ontologies and knowledge representation, and many have suggested this is the solution to semantic interoperability. WordNet is utilized as a knowledge source to build similarity scores between words. This information allows us to identify semantic relationships between data layers. However, we do not use any other ontological sources, or map between data layers and ontologies. Ontologies, the Semantic Web, Linked Data, and why we limit our methodology to word similarity scores, are discussed in detail.

2.7.1 Ontologies

The world we live in is perceived and understood by human observers. From this perception the notion of a conceptualization is introduced. A conceptualization is a set of ideas, theories and beliefs we have regarding some section of the world, be it abstract, physical, or both. Using this definition, an ontology is defined as, "... an explicit specification of a conceptualization" [25]. The term explicit specification denotes some kind of symbolization or encoding, which makes an ontology a tangible entity. This often takes the form of a common vocabulary to represent knowledge. Often, an ontology is more than just a common vocabulary, it consists of defining objects, their properties or attributes, how objects relate to one another, and more [26]. One proposed way of describing an ontology is that it consists of concepts (also known as classes), relations (properties), instances and axioms and hence a more succinct definition of an ontology as a 4-tuple

$$\langle C, R, I, A \rangle \tag{2.3}$$

where C is a set of concepts, R a set of relations, I a set of instances and A a set of axioms [27].

In computer science the term ontology has garnered much interest, and as such as been used in a somewhat vague way [28], and sometimes considered a buzzword [29]. For this work, the purpose of an ontology is that (1) it represents knowledge about some domain and (2) it is explicit. As mentioned in [26], it is not the vocabulary of the ontology that matters, but the conceptualizations that the vocabulary attempts to capture.

The study of ontologies comes originally from epistemology, a branch of philosophy. Epistemology is the theory of knowledge, and is concerned with a variety of questions about knowledge and related topics [30]. The discussion will focus on ontology and knowledge in the context of computer science and Artificial Intelligence (AI).

Ontologies fit into the realm of Knowledge Representation (KR) and Knowledge Management (KM), which are branches of AI. Knowledge is more than a collection of facts because it contains information and also the ability to apply reasoning. The reasoning or ability to perform reasoning is the *intelligence* we strive for in KR. "Description Logics (DLs) are a family of knowledge representation languages that can be used to represent knowledge of an application domain in a structured and formally well-understood way" [27]. DLs describe content with descriptions, which are logical expressions built from atomic concepts and atomic roles. A Knowledge Base (KB) is sometimes used to refer to a dataset with some formal semantics. Although they store data along with axioms, definitions, rules, they differ from ontologies in that they do not attempt to represent a conceptualization. The semantics of data refer to the meaning of data.

Metadata is different from these topics as metadata does not necessarily contain knowledge. As an example, metadata may describe the year the data was created, but provides no uniform or intelligent way of expressing that information to some intelligent agent. It is only useful when humans interpret metadata to discern its semantics.

The relation between ontologies and data modeling must be discussed. One could argue that a database schema is some form of ontology. Considering the process of modeling data, the authors of [31] state “The key distinction between ontology and data modeling is that the former aims to develop general taxonomies of what exists, while the latter aims to develop classifications within a particular application domain”.

Ontologies are sometimes categorized as an upper ontology or top-level ontology. These ontologies are special because they attempt to describe knowledge at high levels of generality. An upper ontology is often used as a general purpose framework for describing relationships and properties, general to various domains of knowledge. It is often combined with a domain specific ontology; the general categories of properties and relationships found in an upper ontology are separated to promote reuse [32].

Some ontologies are referred to as lightweight ontologies. While these ontologies still fit the definition of an ontology, they focus on minimal terminological structure. Such ontologies are taxonomies, which consist of a set of concepts and hierarchical relationships among the concepts [33]. They are easier and faster to create and deploy, and can be modified much more easily, however they sometimes do not have the same power as other ontologies for the purpose of data interoperability.

The term vocabulary has already been mentioned, in that an ontology provides a vocabulary for the domain of knowledge or conceptualization. The difference between a vocabulary and ontology must be stressed. A vocabulary provides a finite list of terms with an unambiguous interpretation of those terms [34]. Ontologies are different because they also specify rules for combining terms and their relations. An ontology specifies terms with semantics independent of reader and context, which extends a vocabulary with some semantic meaning between the terms. A related term is taxonomy, which is a hierarchical categorization of entities within a domain [34]. It could be argued that a vocabulary or taxonomy is an ontology, as it captures and encodes some ideas about a conceptualization. However, the term ontology, in this context, will be reserved for a specification with more information, particularly between objects, than a vocabulary.

The explicit specification aspect of the definition of an ontology implies the requirement of some method of encoding an ontology. There are a number of ontology specification languages available. Among them are KIF, Ontolingua, LOOM, FLogic, SHOE, XOL, RDF, OIL, DAML+OIL, and OWL [34] [35]. Some of the early examples of ontology representation languages, such as KIF, Ontologingua, and LOOM, were popular in the early 1990s and predated Extensible Markup Language (XML). Most of the modern ontology representation languages leverage XML encoding, notably RDF and OWL. These are the standards that make up the much popularized linked data. Today, due to the popularity of the Semantic Web, discussed in the next section, most modern ontologies are encoded with RDFS and OWL.

There are some projects available that some ontologies and projects are now based on. WordNet [12] is a large lexical database of English. It includes nouns, verbs, adjectives, and these words are linked through conceptual relationships. WordNet itself is not strictly speaking an ontology, but the content and its relationships can be used as one. However, the knowledge contained in WordNet is often used to help construct or build ontologies, such as in [36].

2.7.2 The Semantic Web and Linked Data

The Semantic Web is a term introduced by Tim-Berners Lee [37]. The idea is to structure data so that it can be processed by machines automatically. This differs from much of the current information on the World Wide Web, which consists of HTML pages that are manually navigated and interpreted by people. For intelligent machines to ‘understand’ data, information about the data must be provided, in an encoding understood by machines. The

Semantic Web is about making links [38] so information can be linked together, creating a web that machines can traverse. This is the notion of linked data. The four main points of linked data [38] are:

1. Use URIs as names for things.
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
4. Include links to other URIs, so that they can discover more things.

Uniform Resource Identifiers (URIs) can be used to represent an object, not just the location of a web site. This makes it possible to use HTTP to resolve a given object or thing on the Semantic Web. This makes it easy to link to related data sets, by simply providing URIs for related data. The third point is very important, as some standard way of defining information must be used. The standards of the Semantic Web include Resource Description Format (RDF), RDF Schema (RDFS), and Ontology Web Language (OWL). Because of the popularity of these technologies, they are discussed in detail.

Resource Description Framework

The Resource Description Framework (RDF) is the W3C recommendation for semantic annotations on the Semantic Web. It is a language for representing information about resources in the World Wide Web [39], although RDF can be used to describe generic things. In RDF, information is encoded in triples, called statements,

$$(subject)(predicate)(object)$$

The subject refers to the thing the statement is about, expressed as a URI. The predicate expresses information about the property or characteristic of the subject. The object identifies the value of that property [39]. For example, a valid RDF statement would be

http://sweet.jpl.nasa.gov/2.2/matrPlant.owl#Flora, http://www.w3.org/2000/01/rdf-schema#subClassOf,

http://sweet.jpl.nasa.gov/2.2/matrPlant.owl#Plant

This would be interpreted as the subject Flora is a sub class of Plant.

Note that RDF can be encoded with XML, which is sometimes still referred to as RDF, or as RDF/XML.

RDF Schema

RDF Schema (RDFS) is a RDF vocabulary description language. According to [40], RDFS is the first try for expressing simply ontologies with RDF syntax. RDFS expresses predefined entities such as Class, Resource, and Property to define classes and relationships. Among other definitions, RDFS defines *rdfs:domain*, *rdfs:range*, *rdfs:Literal* [39].

Web Ontology Language

The Web Ontology Language (OWL) is an extension to RDFS and is used to define an ontology. Note that OWL refers to version 1, and W3C has released version 2, which will be referred to as OWL2.

Some of the limitations of RDFS are (1) Local scope of properties, (2) Disjointness of classes, (3) Boolean cominbations of classes, (4) Cardinality restrictions, and (5) Special characteristics of properties [41]. To make RDFS more expressive, OWL has been developed. OWL consists of three different standards, OWL Lite, OWL DL, and OWL Full. Ontologies built with OWL still use RDFS and RDF expressions, but are encoded as XML.

The Simple Protocol and RDF Query Language

The Simple Protocol and RDF Query Language (SPARQL) is a SQL-inspired language for performing queries on the Semantic Web [42]. Linked data available on the web is often accessed through a SPARQL endpoint. It is an official W3C Recommendation, and is part of the standards defined for the Semantic Web.

There are several upper level ontologies in existence. A Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [43] and Suggested Upper Merged Ontology (SUMO) [44] are examples of upper level ontologies. Basic Formal Ontology (BFO) is another upper level ontology that overlaps in its efforts with DOLCE and SUMO [45]. The PROTON (PROTo ONtology) ontology is a lightweight upper-level ontology [46].

2.7.3 How our Methodology Utilizes Ontologies

The true pursuit of semantic interoperability is outside the scope of our work. The preceding section is to help the reader understand how the methodology in this thesis differs from an ontological approach. A lexical database is used as a knowledge source for defining similarity between words. Using measures of word relatedness, algorithms are defined to compare two strings to generate a measure of how much those two strings are related, both on syntactic and semantic similarities.

The authors of [47] designed a system to add intelligence to sensor data. They semantically enable SOS by adding semantic annotations to sensor data and using ontology models to reason over observations. Their system architecture has a SOS front end connected to a SPARQL Query Engine, linked to their knowledge base. To make this system as usable and practical as possible, a bottom up approach is the emphasis that doesn't necessitate large, complex ontologies from domain experts. Only word semantics are used to provide groups of related data layers.

The authors of [48] discuss similarity based information retrieval in a SDI. They discuss the various ways of utilizing an ontology, primarily based on subsumption relationships, to answer query results. The entire methodology relies on some well controlled ontology, as well as the user needing to know how information is organized in the ontology, for some searches.

The authors of [49] talk about overcoming semantic heterogeneity in SDIs. Their system is based on a hybrid ontology approach, where each information system has its own application ontology, and each of these ontologies are based on a shared vocabulary, an upper level ontology. To connect the data sources to the application level ontologies, registration mappings is used. The application ontologies and the registration mappings are not easy to generate, and certainly not provided by all, if any, data providers. Although a solid framework, until service providers integrate the use of ontologies, this system is isolated from many of the real data sources we want to connect to.

Sensor Plug and Play is a proposed infrastructure for connecting sensors to the Sensor Web with minimal human intervention [10]. Their scope is much broader than ours, but they do include a section on semantic matchmaking. Their approach is based on building a semantically-enabled matchmaking and mediation framework for the Sensor Web. They discuss the utility of syntactic metrics for matching, such as the Levenshtein distance. However, their proposed Sensor Plug and Play requires data providers to publish metadata about the sensors. The largest difference between their work and ours is that we based our methodology on data available from data providers today, and focused strictly on a bottom-up approach, while they introduced a mediator for semantic matchmaking.

No other high level ontology is used in this methodology. A strong bottom up approach is emphasized by only using word pair similarities, and as a result this approach is extremely flexible and can be reapplied in a variety of SDI contexts. By subscribing to one ontology, many different translations would need to be created and maintained to all external data sources, which is a difficult problem to solve with a fully automatic solution.

Chapter 3

Methodology

This section covers the entire methodology for automatically grouping similar sensor data layers. For this research, similar sensor layers are defined to have the same or related observed properties. In other words, the sensors measure the same general phenomenon.

An overview of the methodology is shown in Figure 3.1. First, data is extracted from multiple SOS services, and converted into sensor data layers known as Property Layers (PLs). Next, text processing is applied to the text data in the PLs. After text processing, the normalized PLs are used in three different grouping strategies.

First, Property Layer Mapping is performed. A dissimilarity function and a threshold are chosen, and maps are created between PLs based on the value of the function and the value of the threshold. Here, a map between two PLs denotes that the two data layers are similar, in that their observed properties are related in a meaningful way.

In the second scenario, clustering is performed to produce PL clusters. The clustering requires a clustering algorithm, a dissimilarity function, and a threshold. The corresponding PL clusters are evaluated to see if they form logical groupings based on the meaning of the data layers.

Lastly, classification on PLs is performed. Using classes generated from a pre-defined dictionary, maps are defined between the PLs and the classes. This allows us to associate zero to many PLs to a class. This is part of the system design implemented in our previous work [8], except the grouping will be performed automatically.

This chapter is as follows. First, the SOS standard and the corresponding data layers, PLs, are discussed in detail. Next, text processing is introduced, and followed by a discussion of why it is necessary to have dissimilarity functions. WordNet as a semantic resource is then

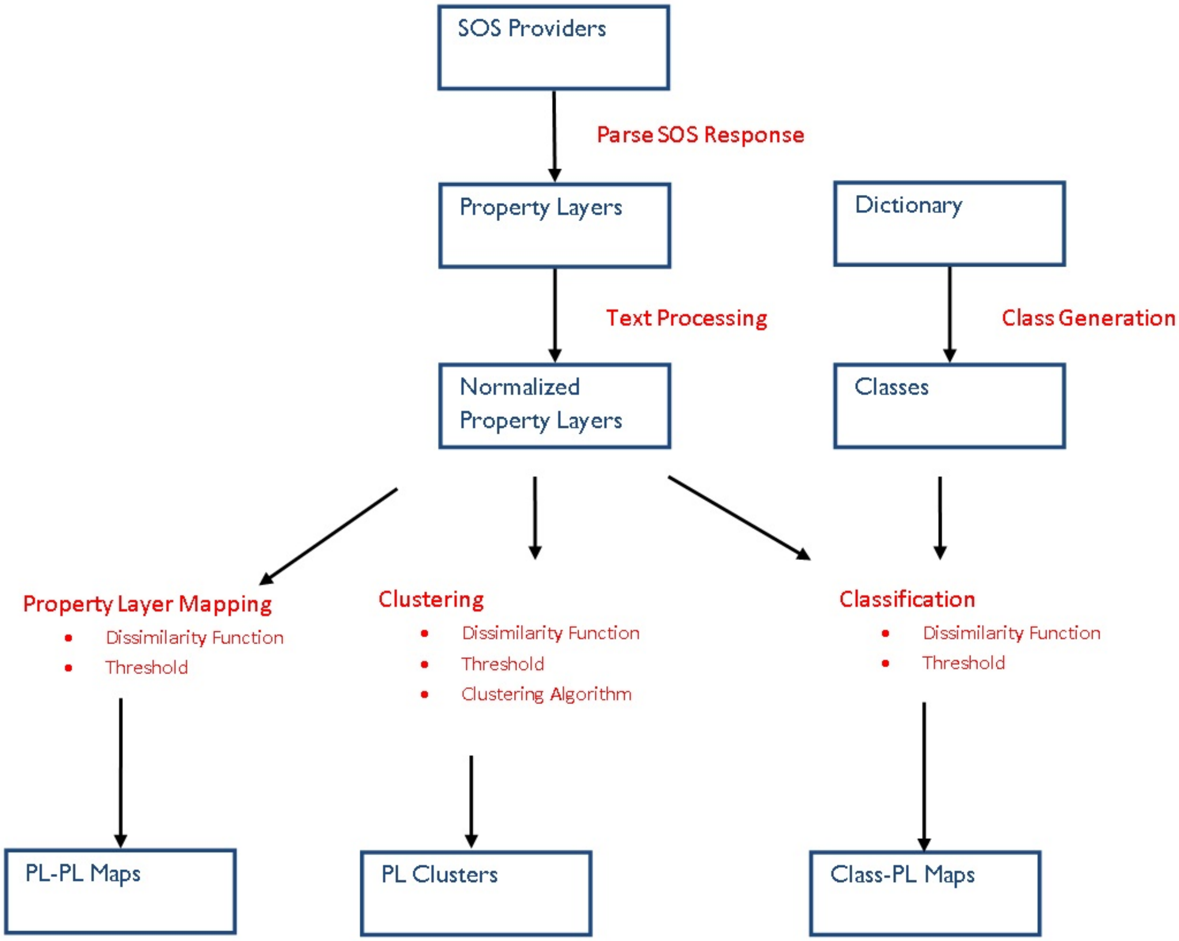


Figure 3.1: General Overview of the Methodology Used to Automatically Group Similar Sensor Data Layers

explained. The notion of dissimilarity functions is explained, and dissimilarity functions are defined. A discussion of the semantic dissimilarity function follows. After that, PL mapping, clustering, and classification are used to automatically group sensor data layers.

3.1 Data

For this thesis, similar data layers are grouped together. A data layer is defined as a group of sensor data readings from a single data provider, measuring a distinct real world phenomenon, live or historical. For this project, data layers will be limited to those available via the Sensor Observation Service (SOS). To understand how exactly data layers are defined, a technical introduction of the SOS standard is necessary.

3.1.1 SOS Overview

SOS is an official OGC standard, and is part of the Sensor Web Enablement (SWE) initiative. It defines how sensor data providers and sensor data consumers communicate to properly encode and send sensor data [50]. SOS runs over HTTP via a client-server architecture. Content is negotiated via XML documents. In this thesis, SOS refers to the SOS standard version 1.0. From hereon, any mention of SOS is an abbreviated form of SOS version 1.0.

In this thesis, a reference to 'SOS' will either refer to the open standards defined by OGC, or in some cases, to a data service that provides sensor data via the SOS standard. In some cases, the phrase 'SOS standard' is used to make it clear we are referring to the standard itself. If SOS refers to a data service, it will always be used with the indefinite article.

Some important terminology from the SOS standard will be used throughout this thesis, and are defined here.

- **Observation Offering** – A logical grouping of data sources
- **Phenomenon** – Some naturally occurring event in the real world that can be measured, (ie. wind speed, air temperature)

- **Observed Property** – Another term for phenomenon
- **Procedure** – A procedure is another term for sensor, except that procedure is more general to include any process that generates an observational value
- **Feature of Interest** (FOI) – An object associated with an observation, such as a lake where a temperature measurement was taken

The SOS standard primarily defines operations, which are requests and corresponding responses of a particular nature. Therefore, when discussing SOS, the operations it provides must be introduced. A SOS must provide three core operations, and there are six other optional operations. For this thesis, these other operations will not be discussed, as they do not contribute any other useful information. The core operations are discussed in detail to understand how data layers are represented in the SOS standard. They are

1. **GetCapabilities**
2. **DescribeSensor**
3. **GetObservation**

GetCapabilities

The GetCapabilities operation is general to all OGC standards. Other OGC standard services also use a GetCapabilities operation to convey all relevant information about that service to a client. The GetCapabilities returns a XML document, which will be referred to as the GetCapabilities file. This GetCapabilities file can be divided into five main XML elements,

1. ows:ServiceIdentification
2. ows:ServiceProvider
3. ows:OperationsMetadata
4. sos:Filter_Capabilities
5. sos:Contents

The elements ows:ServiceIdentification and ows:ServiceProvider provide high level details about the service, such as the service provider's name or the service provider's website.

The ows:OperationsMetadata and sos:Filter_Capabilities elements are important to technical issues of data transfer. For example, they contain the operations and request parameters that are supported by the service.

The element sos:Contents contains a high level overview of the data provided from this service. This is very useful for determining exactly what data is provided by the SOS. The sos:Contents element contains the sos:ObservationOfferingList element. This contains a one to many list of sos:ObservationOffering elements. An observation offering is a "collection of related sensor system observations" [50]. This definition is quite vague, and different SOS implementations utilize observation offerings quite differently. An observation offering may contain only one observed property and one sensor, or it may also contain many different observed properties, sensors, and FOIs.

Table 3.1 shows the information available in a sos:ObservationOffering element. Since sos:ObservationOffering inherits from the Geography Markup Language (GML) abstract feature type, some of the required contents are from GML.

Table 3.1: Details of Element sos:ObservationOffering

Element	Cardinality	Description
sos:time	1	The temporal extent of the data
sos:procedure	0..*	The list of procedures (sensors)
sos:observedProperty	1..*	The list of observed properties. These are used to define unique data layers (PLs)
sos:featureOfInterest	1..*	The list of features of interest
sos:responseFormat	1..*	MIME type of the data that will be returned
sos:resultModel	0..1	Indicates the XML namespace of the response element
sos:respondMode	0..1	Indicates what modes of response are supported
sos:intendedApplication	0..1	The intended category of use for this observation offering
gml:boundedBy	1	The spatial extent of the data
gml:id	1	The id of this observation offering
gml:name	1	The name of this observation offering
gml:srsName	1	The name of the Spatial Reference System

For this research, the focus will be on the sos:observedProperty element. This piece of information is directly relevant to the contents of the data layer. In most cases this symbolizes the name of the data layer.

gml:id/gml:name is the name of the observation offering. This information could be beneficial because sometimes the observation offering name is related to meaning of the data layer. However, it is very inconsistent. The observation offering name may refer to the data provider, the observed property, or even a code or pattern only meaningful to the data provider. In Table 3.2, it is shown how the observation offering may be used to represent different pieces of information. As a result, this information cannot be reliably used in our system.

Table 3.2: Details of Element sos:ObservationOffering

Observation Offering	Observed Property
Setup bureau Bart	urn:ogc:def:phenomenon:OGC:1.0.30:temperature
YSI Multi Water Parameter Sensor 6600ADV University Antwerp	urn:ogc:def:phenomenon:OGC:1.0.30:watertemperature
Temperature	urn:ogc:def:property:ucberkeley:odm:Water Temp C
HT Weather Stations	urn:ogc:def:phenomenon:OGC:airtemperature
ATMOSPHERIC_TEMPERATURE	urn:ogc:def:property:OGC::Temperature
st_denis_evaporation	urn:ogc:def:property:GeoCENS:st_denis_evaporation: AirTemperature

There are also multiple sos:procedure and sos:featureOfInterest elements, creating a list of FOIs and a list of procedures. However, these lists correspond to the observation offering, not necessarily the observed property. Only in the case where an observation offering contains a single observed property can that association be made directly. As with the observation offering, the FOI or procedure name doesn't necessarily describe the feature or the sensor. The name may be a URL, an arbitrary sensor number, or any other string of text that simply can't be used reliably to help identify the observed property. Therefore, the names of the FOIs or procedures are not used to identify relationships between sensor data layers.

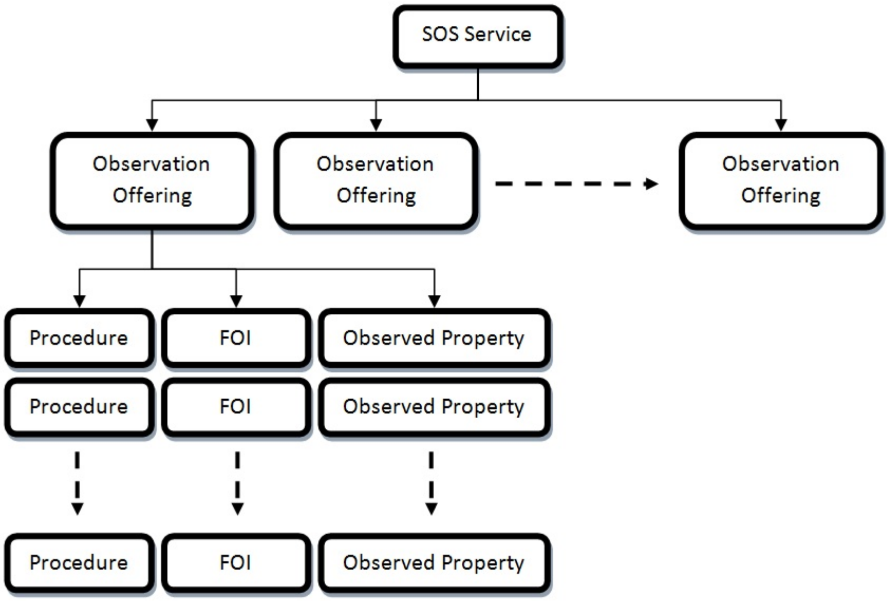


Figure 3.2: SOS Structure Overview

For clarity, the structure of this information is shown in Figure 3.2. Each observed property may have sibling observed properties, with corresponding sibling procedures and FOIs. It is noted that if an observation offering has three observed properties, then it would be logical to assume that those three observed properties are similar in one way or another. However, the SOS standard has been widely interpreted by data providers, as noted in [8]. As such, the sibling relationship between observed properties is not a reliable piece of information to use.

3.1.2 DescribeSensor

The DescribeSensor operation is used to query for more information about a given procedure. Procedure lists are parsed from the observed property. Sensor metadata is encoded in standard known as SensorML [51]. However, many SOS data providers did not return SensorML files based on sensors, and the inconsistency of the SOSs made it impossible to use this information.

3.1.3 GetObservation

The GetObservation operation can be used to collect more information about a PL. "The response to a GetObservation request is an O&M Observation, an element in the Observation substitution group, or an ObservationCollection" [50].

Of these requests, it is particularly commonly to be able to extract is the Unit of Measure (UoM). The UoM is a good indicator of the observations collected, and can be used as an indicator of the meaning of the data layer.

3.1.4 Property Layer

A SOS has the capability to provide a variety of different data about sensors and observations. As a result, our research group has further defined a Property Layer (PL). A PL is a unique data layer, defined by a SOS service URL, an observation offering, and an observed property. Since a SOS, or even a single observation offering within a SOS, may offer a variety of data sources, a PL is the single most atomic data layer available from a SOS.

One to many PLs can be extracted from each SOS. As well, the information relevant to determining the similarity between PLs must be defined. The similarity between PLs is based on how related their observed property is to one another. Therefore, the observed property URI is the single most important piece of information, as it can be considered the 'title' of the data layer. As well, the UoM of the PL will be considered, as it contributes information about the observation collected, based on the observed property.

Although there is a great deal of information available, the inconsistencies of each data provider make it difficult to properly use other pieces of information. A good example of this is the inconsistencies of how observation offerings are used. Table 3.2 shows how the observation offering sometimes refers to the observed property category, which would make that an ideal choice for helping determining how related two PLs are. However, the observation offering label is also used in other ways, and it is outside the scope of this research in differentiating between usable and unusable data names. The observed property URI is the most important and most consistent information that can be used to group similar PLs.

3.1.5 Data Used for this Thesis

For this thesis, data has been extracted from 27 different SOSs. The SOS services were discovered using a Peer-to-Peer (P2P) resource discovery system [52]. There are 212 PLs in our data set. A small example of the dataset is shown in Table 3.3. It must be noted that many of the existing SOS services are in the testing phase, simply due to the gradual development and deployment of the SOS standard. As well, many of the current SOS services online are run by the GeoSensorweb Lab, our own research group. This may cause a bias in the data source, which may influence the evaluation. However, the presented methodology is not influenced by this, and it remains perfectly valid for data collected by future SOS services.

Table 3.3: Example Property Layers

Observed Property URI	UoM
urn:ogc:def:property:geocens: rocky_view_groundwater:groundwater	metres
urn:ogc:def:property:noaa:ndbc:Wind Direction	degree
urn:ogc:def:property:noaa:ndbc:Wind Speed	knot
urn:ogc:def:property:noaa:ndbc:Wind Gust	knot
urn:ogc:def:property:ucberkeley: odm:Solar Radiation Total kW/m^2	kilowatts per square meter
urn:ogc:def:phenomenon:OGC:1.0.30: watertemperature	degC
urn:ogc:def:phenomenon:OGC:waterlevel	m

3.2 Text Processing

As discussed in the previous chapter, text processing is an essential aspect of information retrieval. Two types of text processing are considered, *normalization* and *tokenization*. For edit-based string functions, varying degrees of normalization are investigated to see what effect it has on the results. For set-based string functions, the tokenization and normalization methodology is explained.

3.2.1 Normalization for Edit-Based Functions

A summary of the normalization steps is shown in Table 3.4.

Table 3.4: Example of the Steps of Normalization

Normalization Step	Acronym	Description
Link Stripping	LS	Removes URI prefix
Case Folding	CF	Converts all text to lowercase
Whitespace Removal	WR	Removes all whitespace characters

Link stripping (LS) is the processing of removing the prefix from a URI. The observed property is represented with a URI, in some cases, a reference to an external dictionary. However, the URI prefix often contains little to no semantic information. To isolate the text relevant to the data layer, link stripping is a vital component of text processing. For example, consider the observed property URN, 'urn:ogc:def:property:geocens: rocky_view_groundwater :groundwater'. The entire URI prefix would be identified and removed, leaving the result as 'groundwater'.

Case normalization (CN) is simply covering all uppercase characters to lowercase characters.

Whitespace removal (WR) is the process of removing whitespaces or other characters used for delineating words, such as underscores or commas.

There are some other text processing steps that are worth mentioning, even if they are not used in our methodology. They are mentioned here, as well as a justification for not using them.

Stop words are very common words that do not contribute to the meaning of the text. For example, 'and', 'or', 'if' are words that have caused problems for IR systems. A common text processing task is the removal of these words. However, the text that is processed exists as metadata labels, not free form text, so this step is not necessary for this system.

Stemming is the process of resolving a single word to its root, and isolating its various forms. For example, 'speed', 'speeds', and 'speeding' all refer to the same concept, and stemming involves resolving all three instances to the concept 'speed'. Stemming is not necessary for this project because the text exists as metadata labels, not free form text, although it may be possible to explore stemming as a text processing step in future work.

3.2.2 Tokenization and Normalization for Set-Based Functions

The tokenization and normalization process is explained for set-based functions. Note that a set-based function is simply a function that takes in sets of strings as input. For example, an edit-based function would have two inputs, string one, and string two. For a set-based function, it takes in two lists of strings, where each list would contain one to many strings. Each string in the list is a distinct word, and the differentiation is that a set-based function takes in a series of tokens. For this research, each token is defined as a word, so instead of ‘windspeed’, the input would be [‘wind’, ‘speed’].

First, link stripping is performed to remove all possible URI prefixes.

Next, the string into split into tokens based on whitespace and underscore characters. If this step doesn't tokenize the string, the string is tokenized based on case.

Each token is then further tokenized using the lexical database WordNet. An algorithm is run to attempt to split up a single token into multiple tokens. For example, if the token is ‘windspeed’, it is split into two tokens, ‘wind’ and ‘speed’. Algorithm 1 is shown for completeness.

Algorithm 1 Word Splitting Algorithm

```
INPUT: A string, $s$, of length $N$
Define index $i$, $j$, minimum word size $m \leftarrow 2$
Define list of words, $W$
**if** $N \leq m$ **then**
    Add $s$ to $W$
    Return $W$
**end if**
$i \leftarrow 0$
$j \leftarrow N$
**while** $i < j - m + 1$ AND $i < N - m + 1$ **do**
    $w \leftarrow substring(s, i, j)$
    **if** $isword(w)$ **then**
        Add $w$ to $W$
        $i \leftarrow j$
        $j \leftarrow N$
    **else**
        $j \leftarrow j - 1$
    **end if**
**end while**
**if** $i \neq n$ **then**
    Clear $W$
    Add $s$ to $W$
**end if**
Return $W$
```

Finally, all the tokens are matched to one another. If two side by side tokens form a word, then that word becomes a token. For example, if the resulting list of tokens is [‘battery’, ‘volt’, ‘age’], then first ‘batteryvolt’ is checked as a word. This is not a word, so next ‘voltage’ is checked as a word. Since voltage is a valid English word, the new word list becomes [‘battery’,‘voltage’]. This process is performed iteratively so that the minimum number of valid words are used.

1. Link Stripping is performed, to remove all URI prefixes
2. A string is split up into tokens, based on whitespace, underscore characters and case
3. Each token is split up into smaller tokens using the lexical database WordNet
4. Any two sequential tokens that form a word, according to WordNet, are joined together

3.3 WordNet as a Semantic Resource

In order to define dissimilarity functions that take advantage of the meaning of words, WordNet must first be introduced as a semantic resource. This methodology is similar to the work presented in [53] and [24] where several different approaches are used to generate similarities between words using WordNet. Once similarity values for word pairs exist, then dissimilarity functions can be defined for establishing how similar two PLs are. The following section discusses how WordNet can be used to generate similarity values for word pairs.

WordNet is a lexical network of English words [54]. Nouns, verbs, adjectives, and adverbs are organized into sets of synonyms, or synsets. WordNet is commonly used for its extensive structure of nouns, and as such this will be the focus. The backbone of the noun network is the subsumption hierarchy, consisting of parent-child relationships. The top of the hierarchy contains 11 abstract concepts. Synsets are connected by various relationships, including hyponymy (is-a), its inverse hypernymy, six meroymic (part-of) relations, and antonymy (complement-of). To use WordNet, a global root element is defined such that all the synsets are contained within a graph.

The advantage of WordNet is that it is extremely flexible and not limited to a subset of words, like in an ontology. This ensures that all tokens that are valid English words can be looked up. In this way, the semantic meaning will come from WordNet. The similarity is defined as a value from 0, being not related, to 1, being identical, between two words. The word pair similarity values are generated using various algorithms that run over WordNet.

WordNet has been successfully used in ontology matching [55], in developing more robust algorithms for finding matching classes. As well, these similarity measures have been used to enrich an ontology, as shown in [56].

Several commonly used functions to help describe some of the WordNet based algorithms for generating word pair similarities are defined.

The length between two synsets is denoted $len\left(c_1, c_2\right)$, which is the shortest path between nodes, viewing WordNet as a graph.

The depth of a node is the length of that synset to the global root, denoted $depth\left(c_1, c_2\right)$.

The lowest super-ordinate is the lowest node in the hierarchy that is a parent to both c_1 and c_2. This is denoted $lso\left(c_1, c_2\right)$. For example, $lso(Dime, Credit) = MediumOfExchange$, in the same hierarchy shown in Figure 3.3.

It is also important to note that although synsets are the nodes in WordNet, the relationships we are interested in are between words, or tokens. Because of this, if there are multiple ‘paths’ between two words because the words belong to multiple synsets, the maximum similarity of all the paths is used.

Many of the word similarity approaches do not have formal names, and so we adapt the naming convention from [24]. Table 3.5 is a summary of the abbreviations and approaches used in this thesis. The authors of [24] provide a free tool to calculate word similarities, which is to compute all word pair similarities. As well, we introduce our own basic word pair similarity algorithm, named *ben*. All these approaches are discussed separately based on their general approach.

Table 3.5: Summary of Approaches to Defining Word Relatedness Using WordNet

Approach	Authors	Year	Acronym
Information Content Approach	Resnik [57]	1995	res
	Lin [58]	1998	lin
	Jiang and Conrath [59]	1997	jcn
Path Length Approach	Leacock and Chodorow [53]	1998	lch
	Wu and Palmer [53]	1994	wup
Word Relatedness Measures	Hirst and St-Onge [60]	1998	hso
	Banerjee and Pedersen [61]	2003	lesk
	Patwardhan	2003	vector
Proposed	Knoechel	2012	ben

3.3.1 Information Content Approach

There are three approaches based on information content, *res*, *lin*, and *jcn*. These use a corpus-based measure of the specificity of a concept.

Resnik [57] defines the probability of encountering an instance of a concept, $p\left(c\right)$. The information content is thus defined as $IC\left(c\right) = -\log p\left(c\right)$. The basic idea is that as you move upwards through a taxonomy, the probability of encountering a concept increases. The information content of general, high-level concepts is therefore quite low, because they are related to many other concepts. In the case of a unique top concept, its probability of encountering an instance is one, and its information content is zero. The similarity between a pair of concepts is thus

$$sim_{res} = -\log p\left(lso\left(c_1, c_2\right)\right) \tag{3.1}$$

Which is the information content of the lowest common parent, or *lso*. Concept probabilities were computed by calculating word frequencies using the Brown Corpus of American English. Given the frequency of a concept, the probability is calculated as

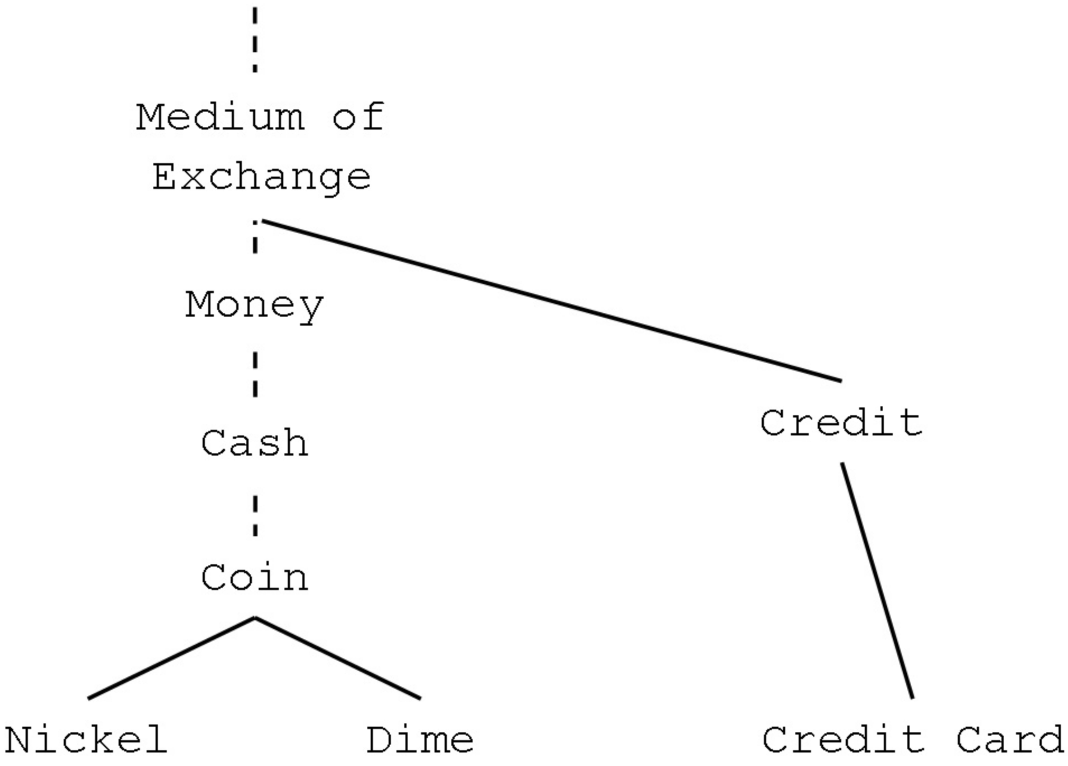


Figure 3.3: Example Hierarchy of WordNet

$$p(c) = \frac{freq(c)}{N} \tag{3.2}$$

The similarity measure proposed by Lin [58] uses the same ideas of information content and probability, however, the measure is different. Lin's focus was that two concepts are similar if they share a close common parent, and not similar if there are many differences between them. The similarity measure is defined as

$$sim_{lin}\left(c_1, c_2\right) = \frac{2\log p\left(lso\left(c_1, c_2\right)\right)}{\log p\left(c_1\right) + \log p\left(c_2\right)} \tag{3.3}$$

Here, it is apparent that the similarity measure is the ratio between the information content of the lowest common parent and the information content of the concepts themselves.

Jiang and Conrath, or *jcn* [59], use an approach very similar to Lin, except the information content is differenced between the lowest common parent and the two concepts. The equation, as a distance, is

$$dist_{jcn}\left(c_1, c_2\right) = 2\log p\left(lso\left(c_1, c_2\right)\right) - \left(\log p\left(c_1\right) + \log p\left(c_2\right)\right) \tag{3.4}$$

The inverse must be taking to convert this distance function to a similarity function.

3.3.2 Path Length Approach

There are two approaches based on path lengths between concepts, *lch* and *wup*.

Leacock and Chodorow's proposed measure [53], *lch*, makes use of the path length. However, the path length is normalized by the maximum depth of the WordNet hierarchy.

$$sim_{lch}\left(c_1, c_2\right) = -\log\frac{len\left(c_1, c_2\right)}{2\max_{c\epsilon WordNet}\left(depth\left(c\right)\right)} \tag{3.5}$$

Wu and Palmer's proposed conceptual similarity [53], *wup*, between a pair of concepts is

$$sim_{wup}\left(c_1, c_2\right) = \frac{2 * depth\left(lso\left(c_1, c_2\right)\right)}{len\left(c_1, lso\left(c_1, c_2\right)\right) + len\left(c_2, lso\left(c_1, c_2\right)\right) + 2 * depth\left(lso\left(c_1, c_2\right)\right)} \tag{3.6}$$

3.3.3 Relatedness Measures

There are three approaches based on word relatedness measures, *hso*, *lesk*, and *vector*.

Hirst and St-Onge's approach, or *hso* [60], classifies relationships in WordNet as having direction. For example, is-a relations are upwards, and has-part relations are horizontal. The shortest path two synsets is computed that is neither too long and doesn't change direction often. *hso* may be summarized by the following equation

$$sim_{hso} = C - len\left(c_1, c_2\right) - k * turns\left(c_1, c_2\right) \tag{3.7}$$

Where C and k are constants, and *turns* is a number of times the path between c_1 and c_2 changes direction.

The *lesk* is a semantic relatedness measure that is based on the number of shared words in their definitions [61]. A gloss is defined here as an alternative word for the description or definition of a word. An overlap is defined as the longest sequence of one or more consecutive words that appear in both descriptions. The sizes of the overlaps found are squared and added together to arrive at the score for a given pair of descriptions, defined as a score. Using this defined as *score*(), relatedness is calculated between two synsets A and B as

$$sim_{lesk} = \sum score(R_1(A), R_2(B)) \tag{3.8}$$

Where R_1 and R_2 are relations in WordNet, such as hypernym. In this way, scores are summed between the two concepts, as well as scores between directly related concepts of A and B. For example, a score between the parent of A and the concept B is calculated using $R_1 = hypernym, R_2 = description.$

The *vector* measure creates a co-occurrence matrix from a corpus made up of the WordNet glosses, defining each concept as a gloss vector. The relatedness between concepts is found by computing the cosine between a pair of gloss vectors.

3.3.4 Basic Subsumption

The basic subsumption algorithm is proposed by our research group, named *ben*. This is a very simple approach based on only direct relationships between words, and is included here as a reference for evaluating all other measures. In this algorithm, we simply see if two words are related via a parent-child relationship. Many of the previously mentioned algorithms traverse WordNet as a graph to find some measure of relatedness. Our approach differs as we are only interested in direct subsumption relationships between tokens. Words that are directly related via a parent-child relationship are given a similarity of 1.0. All other word pairs are given a similarity of 0.0. This measure captures important relationships between words and ignores all others.

3.4 Dissimilarity Functions

It is important to review the concepts of distance, similarity, and dissimilarity for this work.

The similarity between two objects is a numerical measure of the degree to which the two objects are alike [18]. Similarity is often bound from zero (no similarity) to one (complete similarity).The notion of dissimilarity is introduced, which is a numerical measure of the degree to which the two objects are different. Dissimilarity is often in the same range as similarity, but it is also common for dissimilarity values to range from 0 to ∞.

Using these definitions, idea of distance can be introduced as a specific kind of dissimilarity, with well known properties. Distance functions are defined as any dissimilarity function that satisfies the following axioms:

1. $d(x, y) = 0$ iff $x = y$
2. $d(x, y) \geq 0$
3. $d(x, y) = d(y, x)$
4. $d(x, z) \leq d(x, y) + d(y, z)$

The term dissimilarity will be used over distance, as many of the string functions do not satisfy the triangle inequality (4). Dissimilarity is simply a metric used to determine how different two data objects are. For all dissimilarity functions in this report, zero is an exact match, and any value greater than that indicates the two strings are not similar. Many of the dissimilarity functions are normalized from [0,1].

For edit-based dissimilarity functions, the Levenshtein Distance, Length Adjusted Levenshtein Dissimilarity, Jaro dissimilarity, and JaroWinkler dissimilarity are used. For set-based dissimilarity functions, the Jaccard Dissimilarity and Cosine Dissimilarity are used. Semantic dissimilarity metrics are discussed in the next section.

Table 3.6: Summary of Dissimilarity Functions

Edit-Based	Set-Based
Levenshtein Distance	Jaccard Dissimilarity
Length Adjusted Levenshtein Dissimilarity (LALD)	Cosine Dissimilarity
Jaro Dissimilarity	Semantic Dissimilarity
JaroWinkler Dissimilarity	

The Levenshtein Distance (LD) is a function used to calculate a distance between two strings. This distance function counts the number of additions, subtractions, and substitutions required to traverse from one string to another.

As a basic example, consider the two strings 'cat' and 'late'. One string can be traversed to another by substituting 'c' with 'l', and with an addition of 'e' to the end. Therefore, those two operations make the distance between these two strings 2. The algorithm is easiest to understand by the construction of a table between the two strings. 3.7 shows the two strings. Each number is the number of operations it takes to traverse from one sub string to another. For example, to go from 'la' to 'ca' is only 1 operation. The matrix is constructed by building the first row and first column, and then by adding in numbers by each row, left to right. If the two letters match, then take the minimum number as seen from above, to the left, and the upper left. If the two letters do not match, then take that minimum and add one.

Table 3.7: Example of the Calculation of the Levenshtein Distance between *late* and *cat*

		l	a	t	e
	0	1	2	3	4
c	1	1	2	3	4
a	2	2	1	2	3
t	3	3	2	1	2

The LD metric is a distance, but it will still be referred to as a dissimilarity function, as most of the other functions discussed here are not distance functions.

3.4.1 Length Adjusted Levenshtein Dissimilarity

The Length Adjusted Levenshtein Dissimilarity (LALD) is a modification of the LD to reduce the impact of string length on the dissimilarity between strings. This modification is introduced because of problems with the LD with variable string lengths. Consider the two string pairs shown in Table 3.8.

Table 3.8: An Example Showing the Advantages of the Length Adjusted Levenshtein Distance

String 1	String 2	LD	LALD
opticaloxygensaturation	disolvedoxygenclassic	17	0.74
pH	CO	2	1.0

The first two terms may be related, as they both contain the subsequence oxygen. However, their edit distance is 17. Compare that with next string pair, two different observed properties yet the edit distance is only 2.

The LALD is used to prevent the bias of small edit distances between short strings. This modification divides the edit-distance by the length of the longest string.

$$d_{LALD} = \frac{d_{LD}}{\max\left(|s1|, |s2|\right)} \tag{3.9}$$

The dissimilarity is now 1.0 for the short but unrelated strings, and 0.74 for the longer but somewhat related strings.

Note that due to this modification, the term distance can no longer be used. This is because the LALD does not satisfy the triangle inequality axiom. Consider the following three strings:

[‘airtemperature’,‘soilmoisture’,‘soiltemperature’]

And the LALD between them:

airtemperature,soilmoisture = 0.64

airtemperature,soiltemperature = 0.20

soilmoisture,soiltemperature = 0.40

It is apparent that that the direct distance between airtemperature and soilmoisture is larger than the distance traversed using soiltemperature as an intermediate step.

It is noted though that this modification normalizes the output of the function between [0,1].

The Jaro distance metric is a similarity measure between two strings. Literature often refers to this as a distance, however, it is defined here as a dissimilarity function, as it does not satisfy the triangle inequality axiom. Given two strings, $s1$ and $s2$, first the number of matching characters is counted, m. Matching characters means both strings have the same characters at the same offset. For example, consider the two strings ‘john’ and ‘jon’. The second character of each string is ‘o’, so that would be a matching character. Again, consider the two strings ‘john’ and ‘jon’. The third character of each string is ‘h’ and ‘n’, respectively, even though the ‘n’ in the first string looks like it should match with the ‘n’ in the second string. To account for this, characters from $s1$ and $s2$ are considered matching only if they are not farther than $\frac{\max(|s1|,|s2|)}{2} - 1$. For these two strings, the maximum matching distance is now $\frac{\max(4,3)}{2} - 1 = 1$. Since the ‘n’ in ‘john’ is at offset 4, and the ‘n’ in ‘jon’ is at offset 3, so a total of 3 matching characters would be counted between these strings.

Next, the number of transpositions, t, is counted. A transposition is when two letters, side by side, switch positions in one string to produce a matching substring in the second string. For example, consider the two strings ‘receive’ and ‘recieve’. There would be 2 transpositions in total, as the ‘i’ and the ‘e’ in the second string are swapped to produce the first string.

Using both m, t, the number of matching characters and the number of transpositions, respectively, the Jaro dissimilarity is defined as

$$d_{jaro} = 1 - \frac{1}{3}\left(\frac{m}{s1} + \frac{m}{s2} + \frac{m - 0.5 * t}{m}\right) \tag{3.10}$$

This function is modified from being a similarity measure to a dissimilarity measure, by differencing the Jaro similarity from one.

The JaroWinkler dissimilarity is an extension of the Jaro dissimilarity function [15]. This modification uses a prefix scale which gives higher weightings to strings that match from the beginning for a set prefix length. The function is modified such that zero is a match and 1 is no similarity.

$$d_{jw} = d_{jaro} - l * p * d_{jaro} \tag{3.11}$$

Where l is the number of common prefix characters between the strings. In other words, l is the longest substring common to both strings, starting at the first character. p is a constant, a value between 0 and 1. Here, the commonly accepted value of p is 0.10, and l is often limited to a maximum value of 4 [15].

3.4.2 Jaccard Dissimilarity

The Jaccard coefficient is a measure of similarity between two data objects. Given two objects, the Jaccard coefficient is the number of shared binary attributes divided by the total number of binary attributes of both data objects. Therefore, this function requires the input of an array of tokens, where each token is a string. To use this as a dissimilarity function, the value used is simply the difference between one and the Jaccard coefficient.

$$d_{jaccard} = 1 - \frac{m_{11}}{m_{11} + m_{10} + m_{01}} \tag{3.12}$$

In the context of this work, m_{11} is the number of words that exists in both strings, m_{10} is the number of words that exist only in string 1, and m_{01} is the number of words that exist only in string 2. For example, consider two observed property URI strings, after tokenization:

1: [‘air’,‘temperature’]

2: [‘water’,‘temperature’]

The total number of words would be [‘air’,‘water’,‘temperature’], and $m_{11} = 1$, $m_{10} = 1$, and $m_{01} = 1$. Therefore, $d_{jaccard} = 1 - \frac{1}{1+1+1} = \frac{2}{3}$.

3.4.3 Cosine Dissimilarity

The cosine similarity is a very commonly used similarity metric, also known as the dot product. The cosine dissimilarity function is defined as

$$d_{cosine} = 1 - \frac{A \cdot B}{|A|\,|B|} \tag{3.13}$$

Where A and B are vectors of tokens for each input. For example,

1: [‘air’,‘temperature’]

2: [‘water’,‘temperature’]

The length of each vector would be [‘air’,‘water’,‘temperature’]. So consider vector A, where the length would be calculated as $|A| = \sqrt{(1)^2 + (0)^2 + (1)^2} = \sqrt{2}$, because it has a value of 1 for every token it contains, and zero for every other token. Similarly, $|B| = \sqrt{2}$. The dot product would be computed as $A \cdot B = (1 * 0) + (0 * 1) + (1 * 1) = 1$ Therefore, $d_{cosine} = 1 - \frac{1}{\sqrt{2}\sqrt{2}} = 0.5$.

3.4.4 Semantic Dissimilarity Functions

One main issue with the dissimilarity functions described thusfar is that they do not identify semantic relationships between strings or tokens. In the following section, a semantic dissimilarity function is introduced so that semantic relationships between tokens can be established.

This semantic dissimilarity function will be applied between tokens, using the set-based dissimilarity approach. The reason is that arbitrary strings don't necessarily belong to the lexicon of the English language. For example, in edit-based functions, a sample input could be 'windspeed'. The input should be viewed as a string of characters, or the list ['w','i','n','d','s','p','e','e','d']. This entire sequence of characters is not a valid English word, and the only way to evaluate the meaning of the words within the text is to perform tokenization. Obviously, the set-based approach is necessary.

The word pair similarities generated from WordNet, described in the previous section, are used in this dissimilarity function. The dissimilarity function here is very similar to the jaccard dissimilarity measure, which is presented here again as

$$d_{jaccard} = 1 - \frac{m_{11}}{m_{11} + m_{10} + m_{01}} \tag{3.14}$$

The denominator contains the total number of distinct tokens in both token lists. For example, consider two arrays of tokens

A: ['X', 'Y', 'Z']

B: ['X', 'W', 'V']

There are 5 distinct tokens, and a single matching token, so that the dissimilarity would be $d_{jaccard} = 1 - \frac{1}{5} = 0.8$. Assume that word pair similarities are defined between these tokens. This information is used to combine tokens into similar token pairs. To continue with the example for the semantic dissimilarity, it is apparent that 'X' from A is the same token as 'X' from B. Now there are four distinct tokens left, 'Y', 'Z', 'W', and 'V'. A matrix is created of the word pair similarities, and ranked from most similar to least similar.

Table 3.9: Word Pair Similarity Scores for Semantic Dissimilarity Function Example

Token A	Token B	Similarity
Y	W	0.8
Y	V	0.2
Z	W	0.1
Z	V	0.1

It is shown that Y and W have the highest similarity. It is assumed that these two tokens are related, and these two tokens will form a single token pair, 'YW'. The other two tokens, 'Z' and 'V' are also assumed to be related, and form a token pair, 'ZV'. The total number of distinct tokens is now 3, and they are
['X', 'YW', 'ZV']

However, A does not contain 'YW', it only contains 'Y', which is only 0.8 of the pair 'YW'. The dissimilarity function is modified to be

$$d_{semantic} = 1 - \frac{m_{11}}{|A| + |B| - m_{11}} \tag{3.15}$$

Where m_{11} is the total sum of the token pair similarities. This works out to be $m_{11} = 1.0 + 0.8 + 0.1 = 0.9$. Therefore, the overall semantic dissimilarity is $d_{semantic} = 1 - \frac{1.9}{3+3-1.9} = 0.54$.

3.5 Property Layer Mapping

The first of the three major scenarios for this methodology is to define maps between PLs. A map is a symbolic link between two PLs, and the existence of a map between two PLs indicates that those PLs are similar. Two data layers are similar if their observed properties have a direct relationship. Ultimately, PLs must be grouped together in a way that scientists would consider the groups to be intuitively useful for searching sensor data.

A set of maps is collectively referred to as mapping. Therefore, it is important to define mapping between PLs. A PL cannot map to itself. A map is bidirectional. It has no value, it either exists or it does not.

The methodology to define Property Layer Mapping requires two things, a dissimilarity function and a threshold. The process is very straightforward. Every PL is compared to every other PL that is not itself. If the value of the dissimilarity function is less than the value of the threshold, then a map is defined between the two PLs, otherwise nothing happens. For clarity, this is shown in Algorithm 2.

Algorithm 2 Property Layer Mapping

```
INPUT: All Property Layers $P$, threshold $\theta$
Define mapping $M$
for all $p_i$ in $P$ do
    for all $p_j$ in $PLL$ do
     $t = dissim\,(p_i, p_j)$
        if $t \leq \theta$ then
            create map $m = p_i, p_j$
            add $m$ to $M$
        end if
    end for
end for
```

3.6 Clustering

Clustering is performed to automatically group PLs into discrete, non overlapping clusters. The input for clustering is a clustering algorithm, a dissimilarity function, and a threshold. Each clustering algorithm uses the threshold and the dissimilarity function, albeit in different ways. Therefore, the actual clustering methodology depends on the clustering algorithm.

For this methodology, three different clustering algorithms were implemented, K-medoids, DBSCAN, and HAC. K-medoids is a well known variation of the K-means clustering algorithm. DBSCAN is a density based clustering algorithm, which is fundamentally a different kind of clustering algorithm. Lastly, HAC is considered a standard document clustering technique [19]. These three different clustering techniques are evaluated to see which algorithm works the best with this data set. Each clustering algorithm creates distinct, non-overlapping clusters. Fuzzy clustering techniques may be more suitable for this application, as a PL is likely to belong to multiple groups. However, the emphasis of this methodology is on dissimilarity functions, and there are several ways to use those dissimilarity functions to group PLs. The focus remains on straightforward grouping techniques, such as the aforementioned clustering algorithms. Each clustering algorithm is discussed in detail in this section.

3.6.1 K-medoids

K-medoids, also known as Partitioning Around Medoids (PAM), is a similar clustering algorithm to K-means [62]. K-means is a commonly used clustering algorithm. It involves the selection of starting points as seeds, and associating every data point or object to each seed, forming clusters. The centroids of each cluster are calculated, and each data object is re-assigned to a cluster based on the new centroids. This is done recursively until the clusters no longer change, or the change is negligible, between iterations.

However, with string-based representations of data objects, it is impossible to compute centroids of clusters. This is easily possible with numeric data, but with nominal data it's impossible. It is possible to use K-means with tokenized inputs, but for the edit-based dissimilarity functions this would not work. Instead, we use the concept of a medoid. A medoid is simply a data object within a cluster that is close to the centroid, and is used in place of a centroid.

K-means must be modified, so that instead of calculating a centroid, a medoid must be calculated. This can be done by choosing the data object that has the lowest cost. In this sense, the cost is the sum of the distances from the data object to all others in the cluster. If a data object p, is one element in a cluster C_k, then the cost is

$$cost\left(p_i\right) = \sum_{p_j \epsilon C_k, i \neq j} d\left(p_i, p_j\right) \tag{3.16}$$

Similar to K-means, the K-medoids algorithm is very sensitive to the initial selection of classes. To ensure an automatic clustering algorithm, this is a top-down implementation, designed to automatically split clusters that contain outliers. After each iteration, when medoids are no longer changing, the cluster with the largest outlier is selected. If this outlier is greater than a split-threshold, then the outlier is selected as a medoid for a new cluster, and the entire process is repeated.

It is important to understand that this implementation simply iterates a set number of times, θ, and the assumption is that the clusters will not change in a meaningful way after any more iterations. Although this could be improved in future work, the convergence rate is extremely high for the clustering and for this implementation is acceptable.

The algorithm for the K-medoids algorithm is shown in Algorithm 3.

The computational complexity of the K-medoids algorithm is $O\left(n^3\right)$. This is very high, but considering the low number of samples this is certainly acceptable. The K-medoids algorithm can be optimized for future work.

3.6.2 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a density based clustering algorithm [63]. DBSCAN is fundamentally different from K-medoids because it can capture irregular or unusual geometric clusters. The two input parameters are minimum number of points and a value that defines a radius. DBSCAN works by going through all data objects, and if a given data object has at least the minimum number of points 'close by' then those points form a cluster. That cluster is expanded by joining points based on the input parameters, the minimum number of points and the radius. This algorithm treats all data points that don't fit into a cluster as noise. However, it is assumed that there is no

Algorithm 3 K-Medoids Clustering

```
INPUT: All Property Layers $PLL$, threshold $t$, split threshold $\theta$
$outlier = t$
Define set of clusters $Clusters$
Create a single cluster $C$
Set first $pl$ in $PLL$ as medoid for $C$
Add $C$ to $Clusters$
while $outlier \geq t$ do
    for $i \leftarrow 1$ to $\theta$ do
        Clear all $pl$ in $Clusters$
        for all $pl$ in $PLL$ do
            find cluster $C_k$ with minimum distance to medoid
            Add $pl$ to $C_k$
        end for
        for all $C$ in $Clusters$ do
            calculate cost of current medoid, $cost\,(m)$
            for all $pl$ in $C$ such that $pl \neq m$ do
                calculate cost of candidate, $cost\,(pl)$
                if $cost\,(pl) \leq cost\,(m)$ then
                    Set pl as new medoid, $m$
                end if
            end for
        end for
    end for
    $outlier \leftarrow$ find maximum outlier, search all clusters
    if $outlier \geq t$ then
        Create new cluster
        Assign outlier to medoid of new cluster
    end if
end while
```

noise, because each data object represents a unique data layer, which is perfectly valid. As such, we set the minimum number of points to one. Therefore, data points by themselves simply form singular clusters. The algorithm is shown in Algorithm 4.

Algorithm 4 DBSCAN Clustering

```
INPUT: All Property Layers $PLL$, radius $r$, minimum Points, $mPs$
Create list $C$ of length $|PLL|$, set elements to 0 to indicate not visited
Set $c \leftarrow 1$, the current cluster index
for all $p$ in $PLL$ do
    if $C(p) = 0$ then                                  ▷ See if point has been visited yet
        $C(p) \leftarrow -1$      ▷ Set point to -1 to indicate it has been visited, but no cluster assigned
        $q \in PLL$, $q \neq p$
        $n \leftarrow$ array of points around $p$, such that $d(p, q) \leq r$
        if $|n| \geq mPs$ then
            add new cluster, $c \leftarrow c + 1$
            for all $p'$ in $n$ do
                                                        ▷ If point not visited yet
                if $C(p') = 0$ then
                    $q \in PLL$, $q \neq p'$
                    $m \leftarrow$ array of points around $p'$, such that $d(p', q) \leq r$
                    if $|m| \geq mPs$ then
                        add all new points of $m$ to $n$
                    end if
                    $C(p') \leftarrow c$
                                                        ▷ If point visited but no cluster assigned
                else if $C(p') = -1$ then
                    $C(p') \leftarrow c$
                end if
            end for
        end if
    end if
end for
```

DBSCAN is by its definition automatic, although the proper selection of the two input parameters can be difficult. As stated, the minimum number of points will always be set to one, and vary the radius r to see the impact on the clustering.

3.6.3 Hierarchical Agglomerative Clustering

Hierarchical Agglomerative Clustering (HAC) is a clustering algorithm [18] which works by either iteratively splitting one large cluster or combining individual clusters, starting with each data object as a cluster. The latter, a bottom-up approach, was implemented. This means a cluster is created for every data object, and clusters are merged one at a time. To determine which two clusters should be merged, an intra-cluster distance metric is needed. For this project, we used the notion of complete linkage.

The complete linkage of two clusters is defined as the maximum distance of all the possible object distances from one the objects in one cluster to all the objects in the other cluster.

$$linkage = \left(\max\left(distance(p_i, q_j)\right) \mid p_i \epsilon C_1, q_j \epsilon C_2\right) \tag{3.17}$$

The lowest complete linkage of two clusters is calculated for all cluster pairs. If the complete linkage value is less than some threshold value between two clusters, then the two clusters are merged and the process is repeated.

This is a very computationally complex algorithm, $O\left(n^3\right)$.

3.7 Classification

Classification is the task of assigning objects to one of several predefined categories. This methodology implements classification by the creation of maps between PLs and classes. These maps are exactly the same as they have been defined in the previous section, although instead of forming relationships between PLs, relationships are formed between PLs and classes. It must be noted that the classification is done without training data. Generally, classification is performed by using patterns in attributes to isolate and identify classes. This is usually done with a training data set and closely matched with machine learning. Some common algorithms include decision trees, Bayesian classifiers, Artificial Neural Networks, Support Vector Machines, and so forth.

Algorithm 5 Hierarchical Agglomerative Clustering

```
INPUT: All Property Layers $PLL$, threshold $t$
Create set of clusters, $C$
for all $pl$ in $PLL$ do
    Create cluster $c$, add $pl$ to $c$
    Add $c$ to $C$
end for
$m \leftarrow t$
Declare two closest clusters, $c_{m1}, c_{m2}$
while $m \leq$t do
    for all $c_i$ in $C$ do
        for all $c_j$ in $C$, $c_j \neq c_i$ do
            $t \leftarrow linkage\,(c_i, c_j)$
            if $tlem$ then
                $c_{m1} \leftarrow c_i$
                $c_{m2} \leftarrow c_j$
                $m \leftarrow t$
            end if
        end for
    end for
    if $m \leq t$ then
        $merge\,(c_{m1}, c_{m2})$
    end if
end while
```

The important distinction to make is that training data is not abundant or necessarily useful for classification. As such, this methodology is more reflective of the work done in ontological alignment [17].

For classification, class generation is discussed first. Next, the process of matching is discussed.

3.7.1 Class Generation

Before classification can be performed, a set of classes must be created. A class is a data object used to represent of a type of sensor data. For example, we could create a class to represent all sensor data layers that measure temperature. We could then label this class ‘temperature’.

For this methodology, the concept of a dictionary is introduced to represent the class set. A dictionary is simply a set of class names and corresponding descriptions. This is useful because it does not require us to carefully design or build relationships between classes, we simply focus on individual classes.

We implemented the dictionary as a set of unique observed properties. Each dictionary entry will be referred to as a dictionary term, which consists of a well-defined name as a URN, and a dictionary term description. Each class name represents some real world phenomena. For this project, the dictionary was manually created. A data processing team went through several different data sets and created dictionary terms based on the observed properties they encountered. Every dictionary term represents a unique phenomenon, such that no two dictionary terms represent exactly the same observed properties. However, it must be noted that dictionary terms are not mutually exclusive; dictionary terms may be closely related or overlap. For example, ’Precipitation’ and ’Rainfall’ are two possible dictionary terms defined with a ’is-a’ relationship.

This dictionary consists of 79 unique observed properties.

3.7.2 Matching

Matching is the process of establishing maps between classes and PLs. These maps make it possible to associate a single class with zero to many PLs. To perform matching, a dissimilarity function and a threshold is required. As well, the inputs are the list of PLs and the list of classes.

The methodology is as such. For every class term and PL, the dissimilarity will be calculated between them. If that value is less than the threshold, then a map will be made for that class and PL. This process will create a many-to-many relationship between the class list and the PL list. This matching is different from the clustering, as a PL is not exclusively contained within a single class. Since there is not guarantee that classes will be mutually exclusive, this kind of robust mapping is necessary.

The class name URN goes through the same text processing as the PL observed properties. The dissimilarity function computes a value for two inputs, the class name URN of the class and the observed property URI of the PL. The match algorithm is shown in Algorithm 6.

Algorithm 6 Property Layer to Class Mapping

```
INPUT: All Property Layers $PLL$, dictionary terms $D$, threshold $\theta$
Define mapping $M$
for all $d$ in $D$ do
    for all $p$ in $PLL$ do
     $t \leftarrow dissim\,(d, p)$
        if  $t \leq \theta$  then
            create map $m = d, p$
            add $m$ to $M$
        end if
    end for
end for
```

Chapter 4

Evaluation and Results

This chapter evaluates the various methods discussed in the methodology for automatically grouping Property Layers (PLs). First, the evaluation metrics used for the evaluation are introduced. Next, the notion of testing data is introduced, as well as how it was collected. The last three sections of this chapter are the evaluation of dissimilarity functions, clustering, and matching, respectively.

4.1 Evaluation Metrics

The evaluation measures used throughout this evaluation section are discussed. The following evaluation metric definitions are taken from [18]. For the evaluation, there are two different groupings of data. The first set of data is referred to as 'ground truth' or simply similar classes. This is whether or not two objects are actually similar to one another. It is impossible to truly know if two data objects are similar, and the concept of similarity is subjective. As such, this truth is approximated by using humans to label whether or not two data objects are similar.

Secondly, the methodology of this thesis is used to determine whether or not two data objects are classified as similar. The machine classified data is referred to as 'classified as similar', as it is unknown if they are actually similar or not.

We now introduce the notion of a confusion matrix in Table 4.1 with four possible categories for every data object pair.

- f_{++}: True Positive (TP), corresponds to the number of similar PL pairs classified as similar
- f_{-+}: False Positive (FP), corresponds to the number of non-similar PL pairs

Table 4.1: Confusion Matrix

		Predicted Class	
		+	-
Actual	+	f_{++}	f_{+-}
Class	-	f_{-+}	f_{--}

classified as similar

- f_{+-}: False Negative (FN), corresponds to the number of similar PL pairs classified as non-similar
- f_{--}: True Negative (TN), corresponds to the number of non-similar PL pairs classified as non-similar

Two widely used metrics for evaluation are introduced, *precision* and *recall.*

Precision is the ratio of true positives to all classified objects. In other words, precision is high (good) when we do not incorrectly classify false objects.

$$p = \frac{TP}{TP + FP} \tag{4.1}$$

Recall is the ratio of true positives to all truly similar objects. In other words, recall is high (good) when we capture all the true relationships.

$$r = \frac{TP}{TP + FN} \tag{4.2}$$

The F-measure is formally known as the F_1 measure, which is the balance between precision and recall

$$F = \frac{2pr}{p + r} \tag{4.3}$$

Here, a high F-measure ensures the best balance between false positives and false negatives, assuming they are equally bad.

The F_2 measure is based on the F_β measure, which is

$$F_\beta = \frac{(1+\beta^2)\,TP}{(1+\beta^2)\,TP + \beta^2 FN + FP} \tag{4.4}$$

Where β is a non-negative real value. The F_2 measure weights recall higher than precision, meaning the identification of true relationships is important and this measure is more forgiving of identifying false relationships. The equation is

$$F_2 = \frac{5TP}{5TP + 4FN + FP} = \frac{5pr}{4p + r} \tag{4.5}$$

These metrics are used throughout the evaluation. The actual PL groups will be used in a recommendation system or a helpful feature in a SDI for easily finding related data. In this context, the true usefulness of the groups will be based on how they impact the usability of a SDI. This is outside the scope of this thesis, so the evaluation will be limited to optimizing the F-Measure.

4.2 Testing Data

To test how well PL pairs are scored, testing data is required. The most effective way to evaluate the automatic system is to compare the results against human scores of how similar PL pairs are. This human generated data will be referred to as the testing data, and will be used as an approximation to whether or not data objects are similar.

All testing data will be generated by a human operator. I have assumed the role of the human operator in generating this testing data. The main purpose of having testing data is to compare how this methodology groups data layers to how people group data layers. Although different people may group data layers differently, it is assumed these differences are negligible. The main focus is to see if this methodology establishes simple and direct relationships (ie. between precipitation and rainfall) and ignores obviously false relationships (ie. between air temperature and tide gauges). Therefore, the actual person who collects data is irrelevant so long as they make reasonable assumptions about the nature of the data layers.

4.2.1 Property Layer Pair Testing Data

A human operator was chosen to rank PLs against all other PLs, giving them a score of how similar they are to one another. There are 212 PLs in this data set, so for every single PL there will be 211 PL pairs that can be generated.

For this testing data set, one user scored 8 PLs to all other PLs. The 8 PLs are shown in Table 4.2.

Table 4.2: Ground Truth Property Layers for Testing

Observed Property URI
urn:ogc:def:property:noaa:ndbc:Dew Point
urn:ogc:def:property:ucberkeley:odm:Rainfall mm
urn:ogc:def:property:GeoCENS:kenaston_soil_mesonet:SoilMoisture
urn:ogc:def:property:ucberkeley:odm:Solar Radiation Total kW/m∧2
urn:ogc:def:property:geocens:rocky_view_groundwater:groundwater
urn:ogc:def:property:universityofsaskatchewan:ip3:airtemperature
urn:ogc:def:property:noaa:ndbc:Wind Direction
urn:ogc:def:phenomenon:OGC:atomsphericpressure

Each PL pair is rated by the human operator with one of the following categories, (1) not related, (2) weakly related, (3) strongly related, or (4) exactly related. Exactly related PLs refer to data streams measuring the same phenomenon. A strongly related score means that a PL pair have observed properties that are directly related to or are related via a child-parent relationship. A weakly related PL pair is two data layers that may be implicitly related or one phenomenon type may impact the other, such as the relationship between rain and soil moisture. The not related score is assigned when two PLs are not related in a meaningful way. The human operator used the observed property URIs and UoM of the PLs to generate these scores.

It must be noted that the human operator used is not a domain expert. Some of the complicated relationships between environmental sensors may be missed due to this.

4.2.2 Class to Property Layer Testing Data

To evaluate the class to property layer mapping, a different testing data set is required. Information about how classes are related to PLs is needed. This data was collected just as described in the previous section, except class names were selected, and all the PLs are ranked according to that class. For this testing data, all PL relationships were collected for six classes. This is shown in Table 4.3, and accounts for a total of 1272 distinct relationships.

Table 4.3: Ground Truth Class to Property Layers for Testing

Class Name
Soil Moisture
Precipitation Amount
Air Temperature
Relative Humidity
Wind Direction
Atmospheric Pressure

4.3 Property Layer Mapping Evaluation

First, the effectiveness of various dissimilarity functions used to generated PL to PL maps are discussed. The evaluation is performed as follows. All PL pairs that, according to the testing data, are not related or weakly related, are considered dissimilar. All PL pairs that are either strongly related or exactly related are considered similar. To differentiate whether or not two PLs are classified as similar or dissimilar, we require a dissimilarity function and a threshold value. A dissimilarity function produces a value for two PLs, and if that value is less than the threshold, the two PLs are classified as similar. If the function value is equal to or greater than the threshold, the two PLs are classified as dissimilar.

The threshold is a completely arbitrary number, and depends entirely on the dissimilarity function used. As such,the PL-PL maps are evaluated for a variety of threshold values, and look for high F-measures for a given dissimilarity function. It does not necessarily matter at which threshold a dissimilarity function accurately classifies PLs, but rather the maximum F-measure that a dissimilarity function produces.

4.3.1 Dissimilarity Function Evaluation

First, we look at precision, recall, F-Measure, and F2-Measure as a function of the threshold. Figure 4.1 shows how the Levenshtein distance performs at different threshold values. For each distinct threshold value, there exists an entire set of PL-PL mappings. Precision is very high with a low threshold, because the 'thematic filter' is very strict, and only exactly matching character sequences are classified. Therefore, every that is classified by the methodology is correct. However, this also causes a very low recall, because many meaningful relationships are not captured. As the threshold increases, the recall starts to increase, and the precision starts to decrease. As this happens, there is an optimal F-Measure where there are many correctly classified PLs and few true relationships not classified. But once the threshold increases too much, the F-Measure and precision drop sharply, as PLs that are not related are being matched to one another. Many of the other figures follow this pattern, and so for evaluating the effectiveness of a dissimilarity function, the one with the maximum F-Measure is best. This is because that *for some threshold* the dissimilarity function correctly groups PLs together.

The edit-based dissimilarity functions are compared in Figure 4.2. Note that for this figure we normalized the threshold for the Levenshtein distance to [0,1], hence NormLevenshtein. It is interesting to note that the Jaro dissimilarity performs the best, and the basic Levenshtein distance doing the worst. For this figure, normalization of the text included LS,CN, and WR. The set-based dissimilarity functions are compared in Figure 4.3, and these dissimilarity functions outperform all edit-based functions, with the exception of the

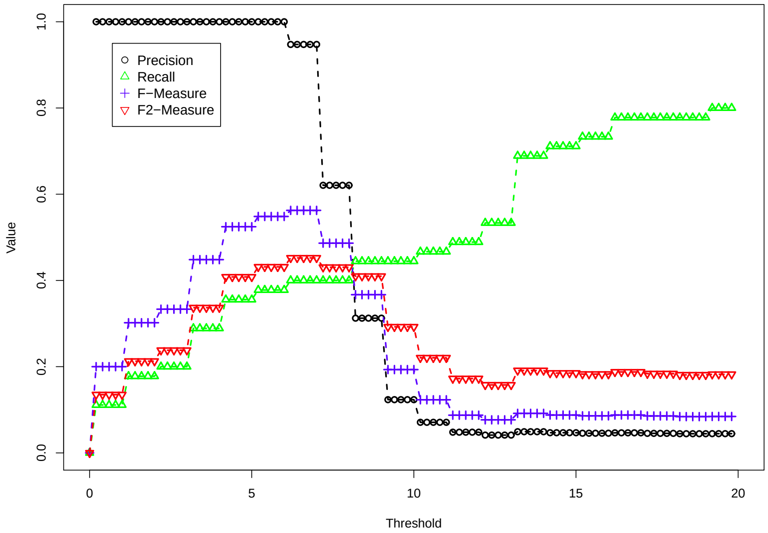


Figure 4.1: Levenshtein Distance Performance

Jaro dissimilarity.

4.3.2 Normalization Evaluation

The impact of normalization is shown in Figure 4.4. Here, we see that the case of the strings doesn't contribute to the semantic meaning of the observed property. In many cases, the whitespace doesn't contribute either, so rigorous normalization is appropriate for the best results with edit-based dissimilarity functions. If we refer to Table 4.4, we see that overall more normalization is better and leads to higher maximum F values.

Table 4.4: Summary of Highest F-Measures of Edit-Based Dissimilarity Functions with Varying Normalization

Tokenization	Levenshtein	LALD	Jaro	JaroWinkler
LS	0.62	0.66	0.70	0.64
LS+CN	0.63	0.67	0.75	0.67
LS+CN+WR	0.61	0.68	0.77	0.68

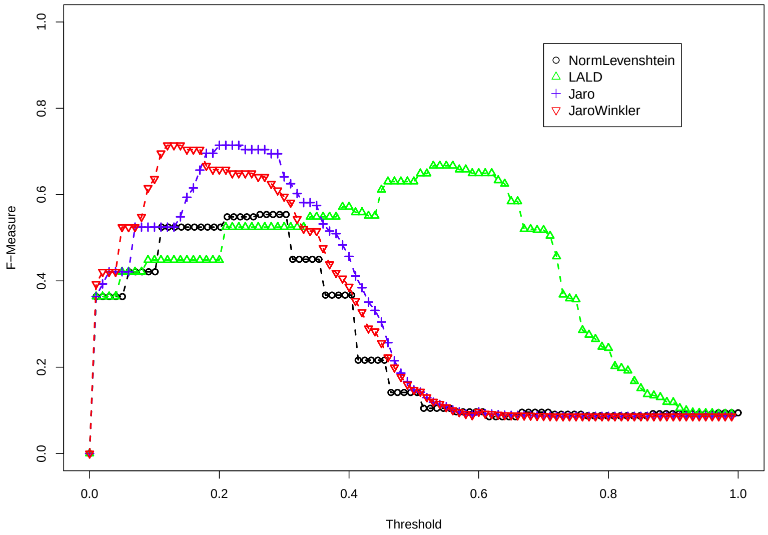


Figure 4.2: Comparison of Edit-Based Dissimilarity Functions

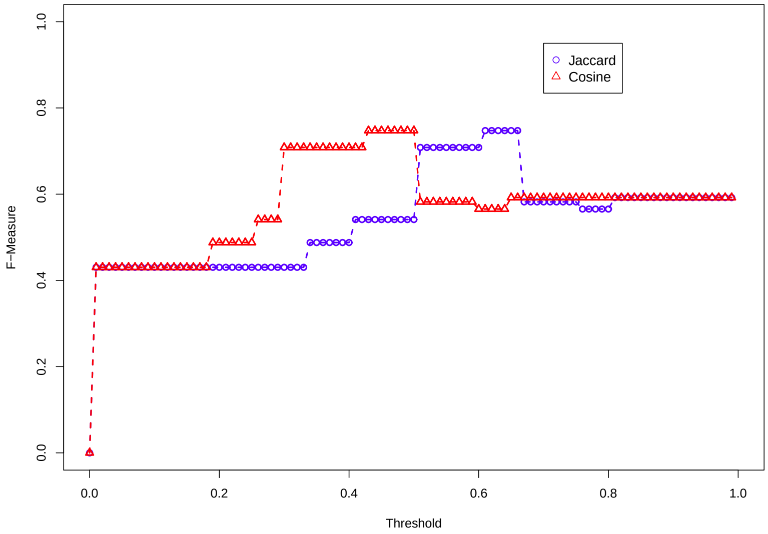


Figure 4.3: Comparison of Set-Based Dissimilarity Functions

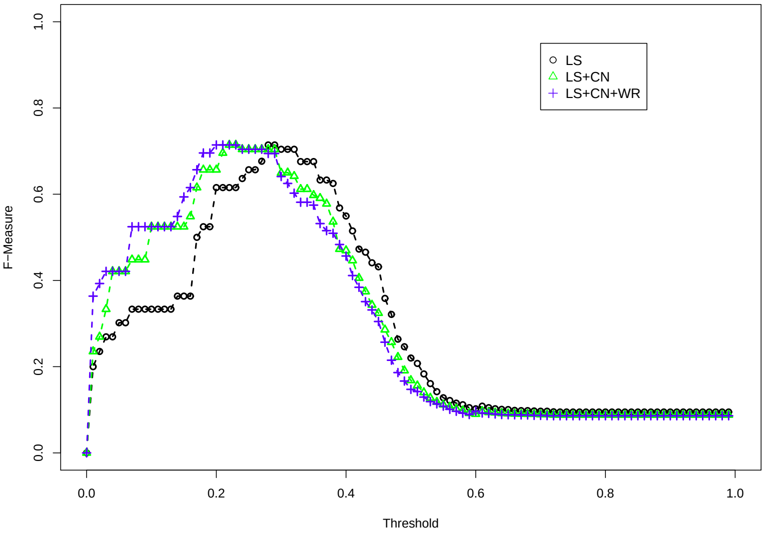


Figure 4.4: Impact of Normalization on the F-Measure for the Jaro Dissimilarity

4.3.3 Observed Property and UoM Evaluation

Next, the evaluate focuses on using the combination of the UoM and the observed property for determining similar PL pairs. To do this, the LALD dissimilarity between observed properties, using LS+CN+WR for normalization. If unit information is missing from one or both PLs, the LALD dissimilarity value is returned. If both PLs have unit information, the unit strings are normalized (LS+CN+WR) and the LALD is used to get a dissimilarity value between the UoMs. The final dissimilarity is equal to

$$d = a * d_{observedproperty} + (1 - a) * d_{UoMs} \tag{4.6}$$

Where $d_{observedproperty}$ is the LALD dissimilarity between the observed properties, d_{UoMs} is the LALD dissimilarity between UoMs, and a is a constant. We make sure that $a + b = 1$, and it is apparent that a is used for simple linear weighting. The results are shown in Figure 4.5.

The UoM does not contribute any useful information, and the performance degrades if

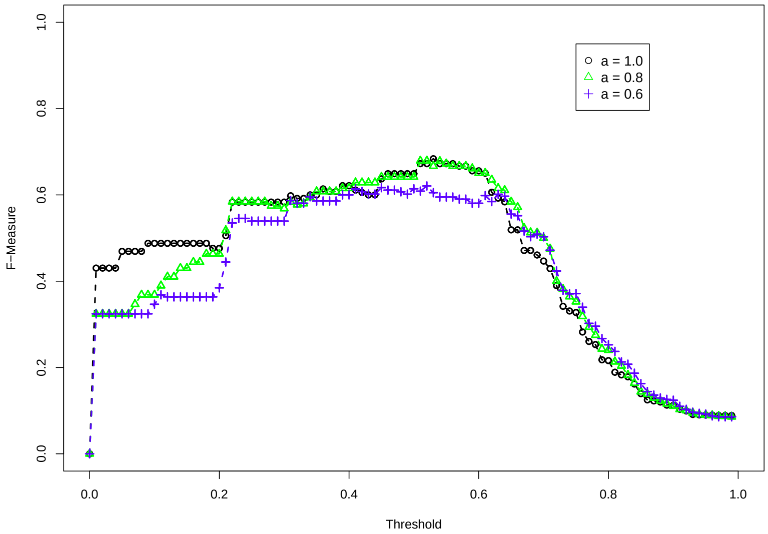


Figure 4.5: Combining the Observed Property URI and the Units of Measure (UoM) in Dissimilarity Evaluation

it is used. This may be because not all PLs had unit information. As well, the UoM shows a high degree of syntax variation. Here basic string similarities simply does not help, as the four strings ‘C’, ‘celcius’, ‘degree celcius’, and ‘Cel’ all refer to the same UoM.

4.3.4 Semantic Dissimilarity Functions Evaluation

Next, semantic dissimilarity functions are evaluated. We only use one semantic dissimilarity function, describe in Chapter 3. We use various word pair similarity values, and compare the effectiveness of those. Since the Jaccard and the Cosine dissimilarity metrics perform at the same level, we will use the Jaccard dissimilarity function for the syntactic only baseline.

We refer to Figures 4.6, 4.7, 4.8, 4.9. The many different WordNet measures are implemented. Figure 4.6 are word similarity measures based on information content. They simply do not perform as well as the baseline Jaccard measure. Figure 4.7 are path lengths between WordNet words, and as well do not perform very well. Figure 4.8 uses three relatedness measures, and these perform very close to the baseline. The *lesk* measure is the only one that performs as well as the baseline. Finally, Figure 4.9 shows a very basic subsumption identification, which slightly but consistently outperforms the baseline.

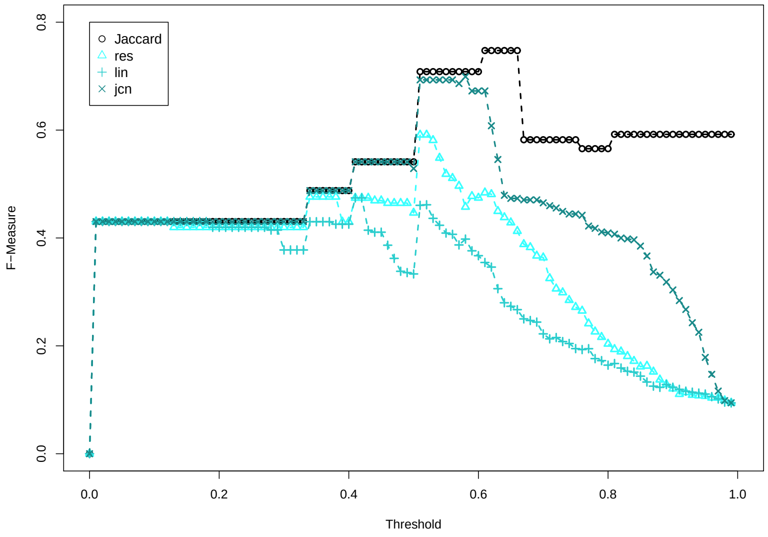


Figure 4.6: Semantic Dissimilarity Evaluation using Word Similarities Derived From WordNet, based on a Information Content Approach

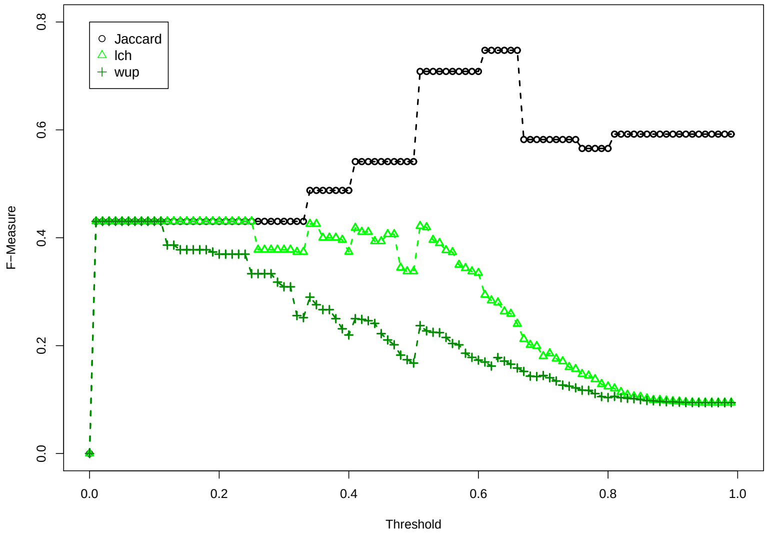


Figure 4.7: Semantic Dissimilarity Evaluation using Word Similarities Derived From WordNet, based on a Path Length Approach

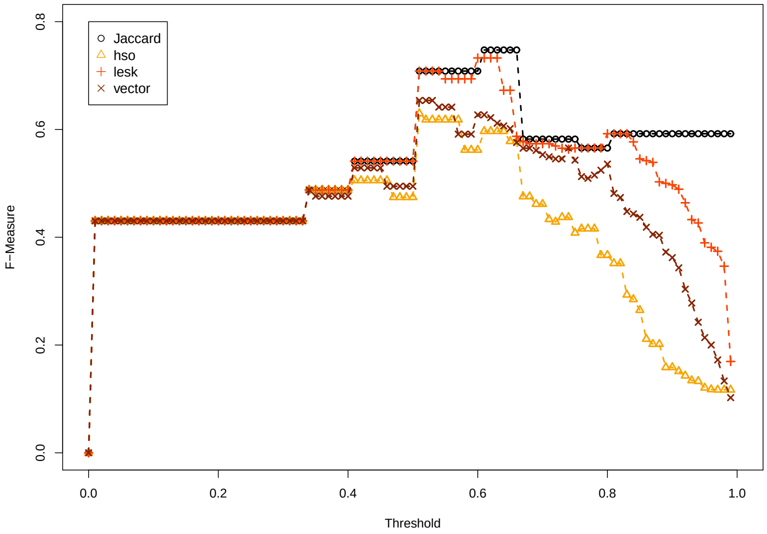


Figure 4.8: Semantic Dissimilarity Evaluation using Word Similarities Derived From WordNet, based on Word Relatedness Measures

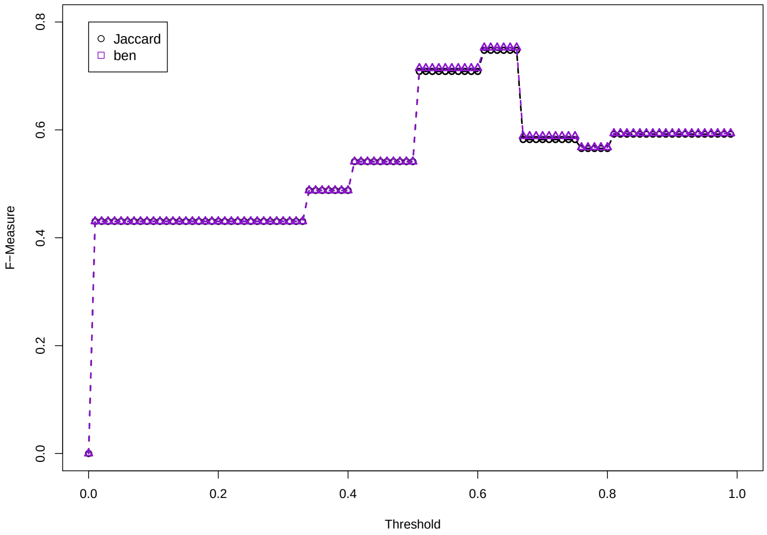


Figure 4.9: Semantic Dissimilarity Evaluation using Word Similarities Derived From WordNet, using the Proposed Approach

This evaluation shows that the semantic dissimilarity function performs consistently behind the basic Jaccard dissimilarity function.

4.3.5 Error Classification

To further understand the PL mapping accuracy, all the false positives and false negatives are classified in a table for a dissimilarity function. This error classification table shows how errors are generated, and where various dissimilarity functions do not perform very well. Error classification is performed for the Jaro dissimilarity function, and presented in Table 4.5. Error classification is performed for the Jaccard dissimilarity function, shown in Table 4.6. Finally, the error classification is presented for the semantic dissimilarity function for *res* word pair similarity scores in Table 4.7.

Table 4.5: Error Classification for Jaro with LS+CN+WR Normalization

Type	Error	Number of Cases	Description
No Error	TP	42	
Semantic Meaning Missing	FN	20	There was no semantic meaning to link the two PLs, for example, dew point and relative humidity
Word Association	FP	5	There was a common word between PLs that gave a false relationship, for example, 'groundwater' and 'groundheatflux'

Table 4.6: Error Classification for Jaccard

Type	Error	Number of Cases	Description
No Error	TP	37	
Semantic Meaning Missing	FN	19	There was no semantic meaning to link the two PLs, for example, rainfall and precipitation
Word Association	FP	12	There was a common word between PLs that gave a false relationship, for example, ‘water temperature’ and ‘ground water’
Abbreviation	FN	3	Common abbreviations were not understood by this system, for example, ‘Temp’ as an abbreviation for ‘Temperature’
Incorrect Tokenization	FN	2	One of the PL observed properties was incorrectly tokenized and as a result, the appropriate tokens were not matched

Table 4.7: Error Classification for Semantic Dissimilarity Using *res*

Type	Error	Number of Cases	Description
No Error	TP	34	
Semantic Meaning Missing	FN	22	The semantic meaning between PLs was not established, or it existed between compound nouns and was not captured by single word similarity scores. For example, dew point and relative humidity
Additional Words	FN	11	There were many truly similar PLs that were missed because of extra words that did not contribute to the semantic meaning, for example, ‘rainfall mm’ and ‘rainfall rate’, the additional tokens increased the dissimilarity value too high
Not Strictly Related	FP	10	Some PLs are related, but the human operator determined that they were not sufficiently related, for example, ‘rainfall’ and ‘hail’
Abbreviation	FN	3	Common abbreviations were not understood by this system, for example, ‘Temp’ as an abbreviation for ‘Temperature’

4.4 Clustering Evaluation

The clustering is evaluated using the property layer pair testing data. PL pairs are classified as similar if they exist within the same cluster. Because a PL will only appear in a single cluster, all PL pairs can be classified as similar by being the same cluster, or classified as dissimilar because they exist in separate clusters. The F-measure is used as a metric to determine the best clustering algorithm. As in the property layer mapping evaluation, two PLs are similar if they have a human assigned score of 2 (strongly related) or 3 (exactly related). If a PL pair has a human assigned score of 0 (not related) or 1 (weakly related), then the two PLs are not related.

The evaluation requires a dissimilarity function, a threshold, and a clustering algorithm. Each clustering algorithm uses the threshold value in a different way, as describe in Chapter 3. The three clustering algorithms, K-Medoids, DBSCAN, and HAC are compared with three different dissimilarity functions. First, clustering with LALD is shown in Figure 4.10. Clustering with the Jaro dissimilarity is shown in Figure 4.11, and Figure 4.12 shows clustering using the set based Jaccard dissimilarity. The three clustering algorithms are compared together for each dissimilarity function.

Although DBSCAN has the best F-Measure, it fundamentally is a poor choice as a clustering algorithm. Density based clusters have the potential to have very high intra-cluster ranges, which is often problematic. It is clear that at a critical point increasing the threshold completely reduces the effectiveness of the clustering.

HAC and K-medoids are both more appropriate. Since HAC uses complete linkage, we see that as the threshold increases, the F-Measure doesn't sharply decrease like in DBSCAN or K-Medoids.

We use the *lesk* word similarity scores in the semantic dissimilarity function for clustering evaluation, shown in Figure 4.13. The semantic dissimilarity function out performs the Jaccard dissimilarity function, although the best is the LALD. However, we can see that the

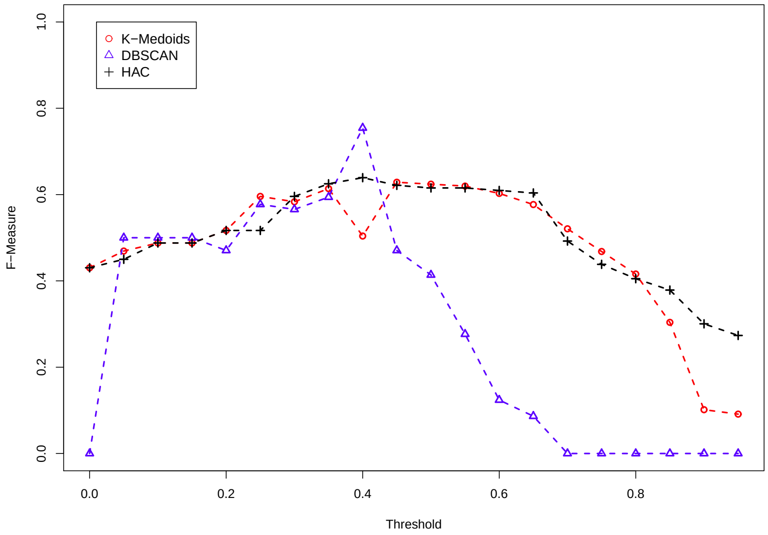


Figure 4.10: Clustering Results Using Length Adjusted Levenshtein Dissimilarity

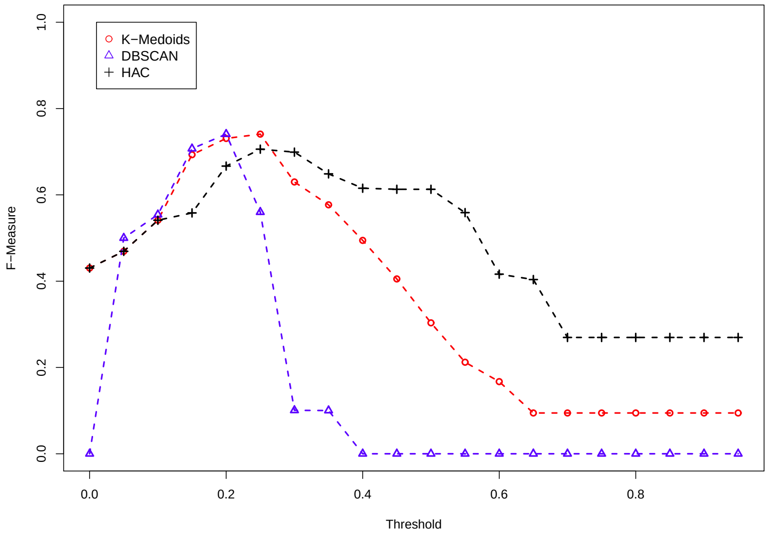


Figure 4.11: Clustering Results Using Jaro Dissimilarity

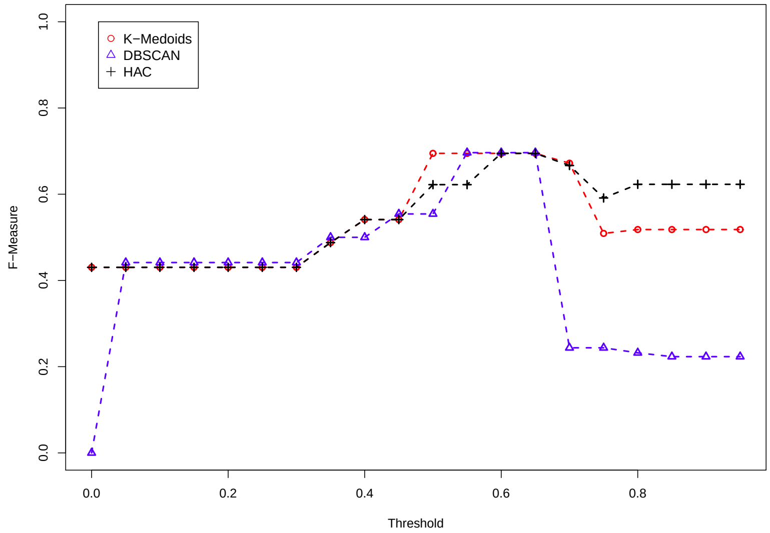


Figure 4.12: Clustering Results Using Jaccard Dissimilarity

most consistent performance comes from the semantic dissimilarity.

Table 4.8: Highest F-Measures for Clustering

Dissimilarity Function	Best F-Measure
LALD	0.75
Jaro	0.74
Jaccard	0.70
lesk	0.74

4.5 Class to Property Layer Mapping Evaluation

The mapping of classes to PLs is evaluated in this section. For this evaluation, the observed property of the PL is compared directly to the class names. The class to property layer mapping evaluation is very similar to the property layer mapping evaluation. The class to property layer testing data contains a score from 1 - 4 for all class-PL relationships. A class-PL pair is similar if the score is a 3 (strongly related) or a 4 (exactly related). A class-PL is

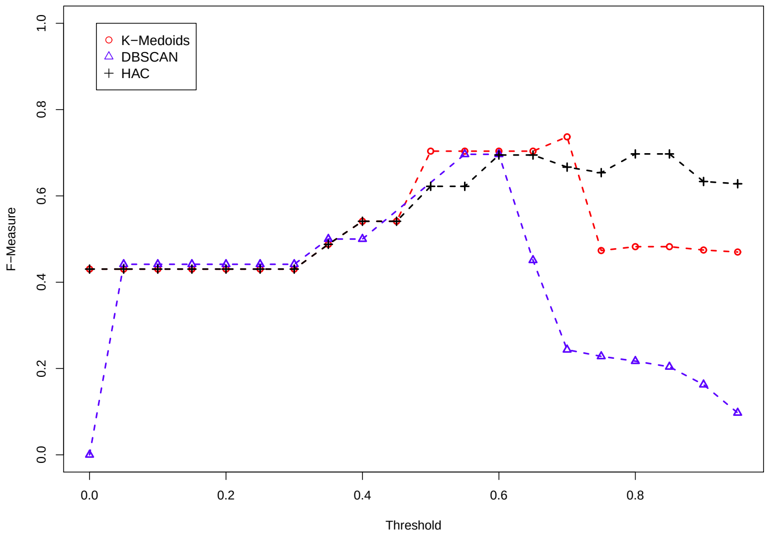


Figure 4.13: Clustering Results Using Semantic Dissimilarity, using *lesk* Word Similarity

dissimilar if the score is 0 (not related) or 1 (weakly related).

For classification, mapping between classes and PLs must be defined. A class-PL is classified as similar if a map is created between both the class and the PL. A class-PL is classified as dissimilar if no map exists between them. To create the mapping, we need to define a dissimilarity function and a threshold. As described in the previous Chapter, a map is created if the value of the dissimilarity function is equal to or below the threshold.

Figure 4.14 shows that the edit dissimilarity functions perform quite well. The JaroWinkler dissimilarity function performs the best, and greatly outperforms the Jaro dissimilarity function.

Figure 4.15 shows that both the Jaccard and Cosine dissimilarity perform at the same level, albeit at different thresholds.

Figure 4.16 shows all the various semantic dissimilarity functions. As before, the dissimilarity functions have lower F-Measures than the baseline, in almost all cases.

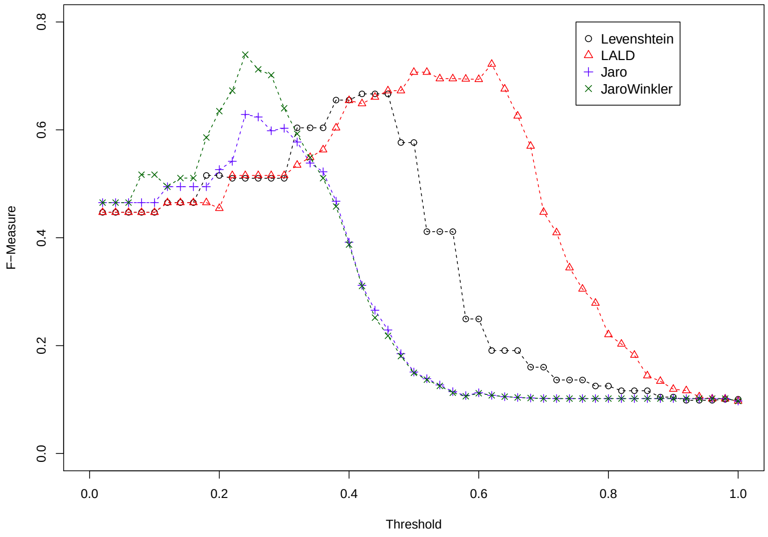


Figure 4.14: Matching Results of Edit Based Dissimilarity Functions

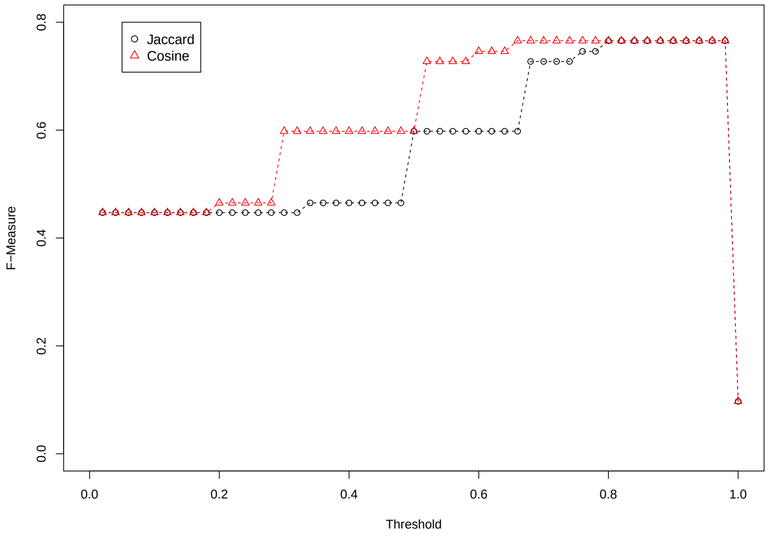


Figure 4.15: Matching Results of Set Based Dissimilarity Functions

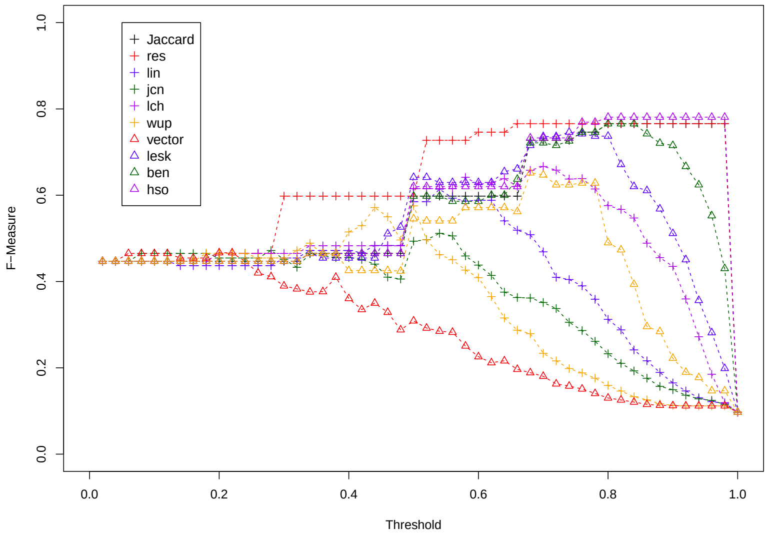


Figure 4.16: Matching Results of Semantic Dissimilarity Functions

4.6 Discussion

The semantic dissimilarity functions were expected to perform much better than the purely syntax based string dissimilarity measures. However, in this evaluation, we see that the semantic dissimilarity function performs quite poorly. It is important to understand exactly why these measures are not performing as well as a basic syntactic measure.

In Tables 4.9, 4.10, and 4.11, we look at three different PL pairs. The word pair similarity scores, generated from WordNet, are shown between all the words of the observed properties. As well, the resulting semantic dissimilarity scores are shown.

Table 4.9: Word Pair Similarity Scores and Semantic Dissimilarity Values Between *ground water* and *wind speed*

PL 1		ground water								
PL 2		wind speed								
word1	word2	res	lin	jcn	lch	wup	hso	lesk	vector	ben
ground	wind	0.29	0.32	0.08	0.53	0.74	0.40	0.03	0.21	0.00
ground	speed	0.29	0.42	0.11	0.53	0.62	0.40	0.02	0.06	0.00
water	wind	0.27	0.24	0.08	0.47	0.67	0.40	0.05	0.04	0.00
water	speed	0.42	0.43	0.12	0.53	0.63	0.40	0.03	0.03	0.00
Dissimilarity		0.78	0.77	0.95	0.64	0.48	0.75	0.98	0.94	1.00
Jaccard Dissimilarity										1.00

Table 4.10: Word Pair Similarity Scores and Semantic Dissimilarity Values Between *ground water* and *soil moisture*

PL 1		ground water								
PL 2		soil moisture								
word1	word2	res	lin	jcn	lch	wup	hso	lesk	vector	ben
ground	soil	0.70	1.00	1.00	1.00	1.00	1.00	0.53	0.78	0.00
ground	moisture	0.07	0.09	0.07	0.40	0.43	0.00	0.00	0.01	0.00
water	soil	0.38	0.63	0.19	0.56	0.71	0.40	0.07	0.07	0.00
water	moisture	0.07	0.09	0.07	0.38	0.38	0.00	0.01	0.02	0.00
Dissimilarity		0.76	0.62	0.64	0.48	0.48	0.67	0.84	0.75	1.00
Jaccard Dissimilarity										1.00

Table 4.11: Word Pair Similarity Scores and Semantic Dissimilarity Values Between *rainfall mm* and *cumulative hail mm*

PL 1		rainfall mm								
PL 2		cumulative hail mm								
word1	word2	res	lin	jcn	lch	wup	hso	lesk	vector	ben
rainfall	mm	0.00	0.00	0.06	0.23	0.20	0.00	0.01	0.01	0.00
rainfall	cumulative	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
rainfall	hail	0.66	0.83	0.30	0.70	0.91	0.50	0.08	0.27	0.00
mm	cumulative	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
mm	hail	0.07	0.00	0.05	0.35	0.35	0.00	0.01	0.01	0.00
Dissimilarity		0.50	0.43	0.65	0.48	0.38	0.57	0.73	0.66	0.75
Jaccard Dissimilarity										0.75

It must be noted that in these tables, the biggest problem the semantic dissimilarity functions have is that they always have lower dissimilarity values than the baseline. It makes sense that some words are related, for example, rainfall and hail. However, some words are closely related, and it doesn't always seem important, such as water and soil.

One way to show how these functions perform is to list all PL pairs, using a single PL as a reference. Using the PL defined by the observed property, 'wind direction', all matching PLs are listed using a syntactic measure, Jaccard, and a semantic measure, using word similarities defined by *res*. This is shown in Table 4.12. It is shown that the semantic measure matches wind direction to dew point, mean wave direction, and other observed properties that are not directly related.

This table shows how a semantic similarity measure can give quite low dissimilarities for seemingly unrelated PLs. The reason other semantic similarity measures perform better than others is that they assign lower similarities between words, which results in less false negatives.

We can conclude that due to the expansive nature of words in general, semantic similarities exist between words that give unintended relationships and meaning where there shouldn't be any. However, it is still important to use these functions as there is no way to

Table 4.12: Similar PLs to the PL Defined by Observed Property *wind direction*, Comparing Jaccard and Semantic Dissimilarity Functions

Jaccard		res	
WindDirection	0	WindDirection	0
winddirection	0	winddirection	0
winddirection	0	winddirection	0
Wind Direction Degrees	0.33	Wind Direction Degrees	0.33
direction	0.5	direction	0.5
Wind Gust	0.67	windrun	0.53
Wind Speed	0.67	WindSpeed	0.54
WindSpeed	0.67	windspeed	0.54
windspeed	0.67	windspeed	0.54
windspeed	0.67	Wind Speed	0.54
windrun	0.67	windspeed	0.54
windspeed	0.67	windspeed	0.54
windspeed	0.67	Mean Wave Direction	0.54
Mean Wave Direction	0.75	Dew Point	0.55
Wind Speed Avg MS	0.8	dewpoint	0.55
Wind Speed Max MS	0.8	dewpoint	0.55
		dewpoint	0.55
		Wind Gust	0.64
		Pressure Tendency	0.65
		snow depth	0.66
		frost depth	0.66
		snow depth	0.66
		snow depth	0.66
		snow depth	0.66
		thaw depth	0.66

bridge concepts like precipitation and rainfall without some use of a knowledge source.

Chapter 5

VirtualSOS

In the previous chapters, a methodology has been presented for automatically grouping sensor data layers, which we call Property Layers (PLs). In this chapter, a system called **VirtualSOS** is presented. VirtualSOS uses the methodology described in Chapter 3 for grouping similar sensor data layers, and integrates it into a web service. Note that VirtualSOS is not a SOS service, nor does it adhere to the SOS standard. VirtualSOS is an online platform that showcases the concept of grouping data layers by their observed properties to facilitate data browsing, connecting to active SOS services. The purpose of this application is to allow the user to select a single, unique 'virtual' layer, which is built up of one to many real, physical data layers. In this chapter, VirtualSOS is introduced and discussed as a proof of concept software component that provides a similar data layer grouping service to support SDIs.

To understand VirtualSOS, the logical construction is first discussed. Next, a more structural description is provided, along with screenshots of the front end.

The VirtualSOS applications consists of the following logical components. Their relationships are shown in Figure 5.1.

1. A list of PLs, extracted from SOS services
2. A dictionary, which is a unique list of observed properties
3. A list of mappings from dictionary terms to PLs
4. A set of taxonomies used for organizing dictionary terms

The Property Layers (PLs) are extracted from a pre-generated SOS list, as described in Chapter 3. These PLs will consists of the physical data layers.

The dictionary is a list of unique observed properties. Each dictionary entry will be

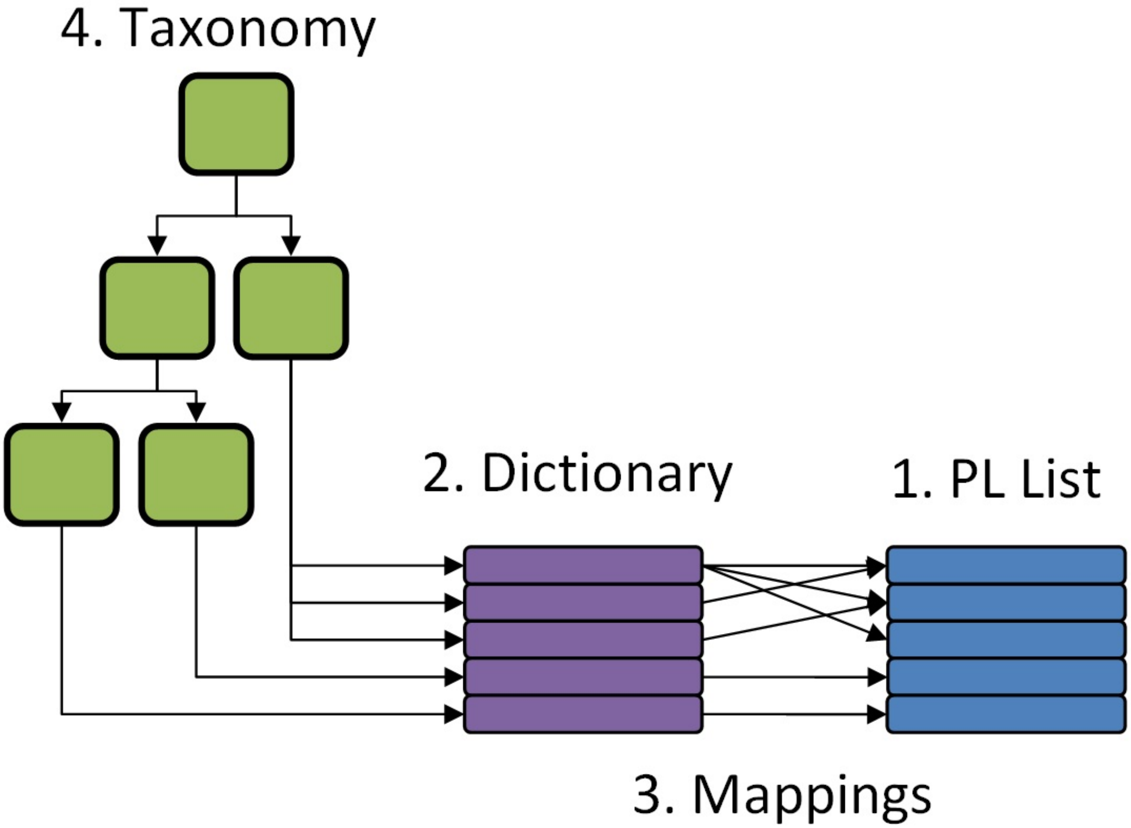


Figure 5.1: VirtualSOS, a Web Based Application of Grouping Similar Sensor Data Layers

referred to as a dictionary term, which consists of a well-defined name as a URN, and a dictionary term description. For VirtualSOS, the dictionary was manually created. A data processing team went through several different data sets and created dictionary terms based on the observed properties they encountered. Every dictionary term represents a unique observed property, such that no two dictionary terms represent that same observed property. However, it must be noted that dictionary terms are not mutually exclusive; dictionary terms may be closely related or overlap. For example, 'Precipitation' and 'Rainfall' are two dictionary terms defined with a 'is-a' relationship. The relationships between dictionary terms are represented implicitly in taxonomies (discussed below). However, this is strictly for human interpretation, and is not used in any way by the system. This dictionary is used as a virtual layer list, so the terms 'dictionary term' and 'virtual layer' refer to the same entities.

Every dictionary term has a ranked list of corresponding PLs. This is generated using the classification methodology described in this thesis, treating dictionary terms as classes. However, it is slightly different from the mapping defined earlier. For VirtualSOS, a map is a dissimilarity value from zero to one, denoting the relevancy of the class-PL relationship. In Chapter 3 and 4, we have defined a map as a boolean relationship between a class and a PL. The creation of valued maps utilizes the exact same methodology as described earlier, except the value of the dissimilarity function between a class and a PL is simply the value of the map. The reason valued maps were chosen for VirtualSOS is that users will be allowed to select their own threshold, and automatically select all PLs that were below that threshold.

Taxonomies are used to organize the dictionary terms to facilitate data browsing. Every dictionary term is the leaf term in the taxonomy, and all non-leaf nodes are any concept represented by text. For example, a non-leaf node could be 'Atmosphere', and could contain PLs that are measuring phenomenon in the atmosphere. In addition, the system is designed to support multiple taxonomies. This is useful because many different users from different

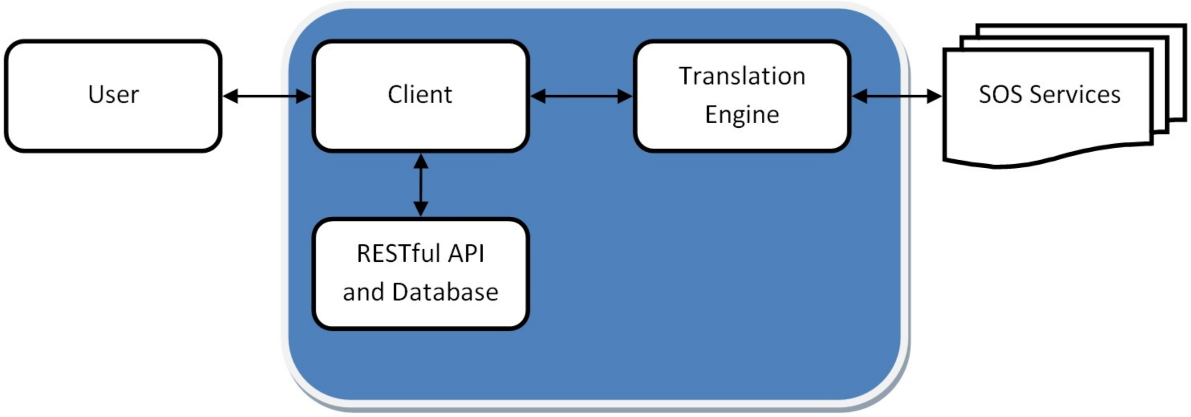


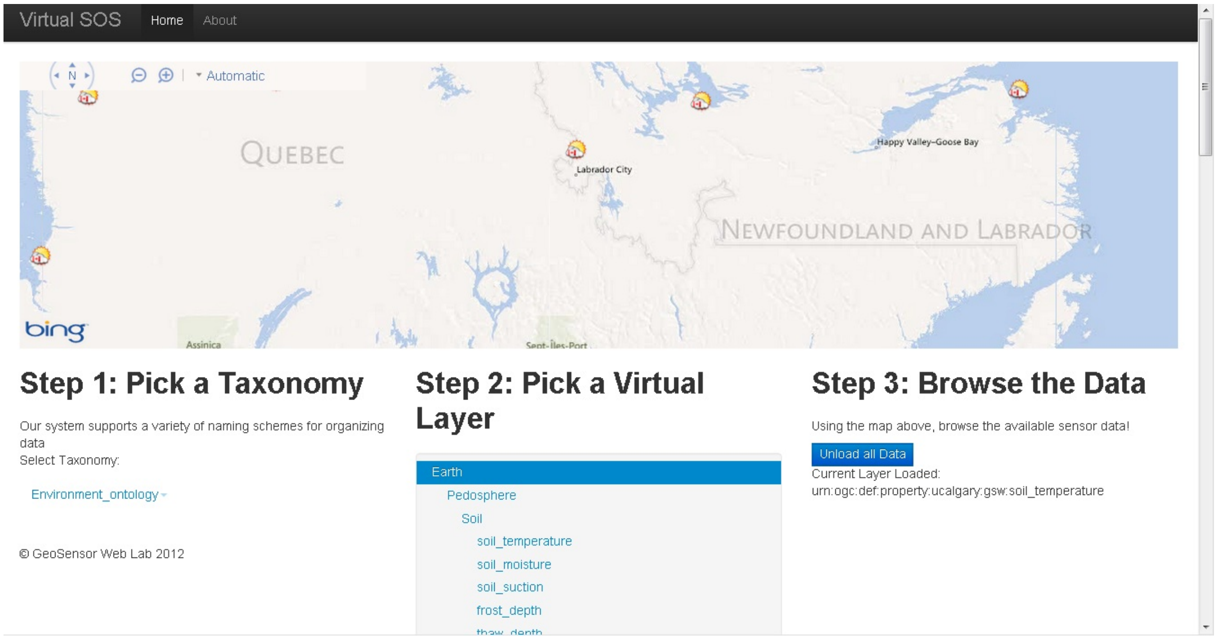
Figure 5.2: VirtualSOS Architectural Diagram

domains will likely have different ideas on how to organize data layers. Support for multiple taxonomies allows various domain-specific taxonomies to exist, and the user may select the taxonomy that is useful to them. As well, the system could easily be extended to support user defined taxonomies, which would greatly increase the usability of the service.

The implementation of VirtualSOS is described to give the reader an idea of the various software components involved.

The implementation of VirtualSOS is based on three major components, (1) a database, (2) a REST-ful API, and (3) a web based front end. We use PostgreSQL as a database and it contains all the data for VirtualSOS, including the PL list, the mapping, the taxonomy, and the dictionary. The REST-ful API was written in Java, on top of the Restlets [1] stack, and communicates with the database to respond to requests via a basic JSON exchange over HTTP. The web based front end was written in HTML, PHP, and Javascript. It communicates with the REST-ful API to retrieve and display data that the user selects, using a BingMaps interface. The front end is also responsible for loading data, however, it communicates with another software component, called the Translation Engine, to do this [8]. The front end is shown in Figure 5.3.

[1] Restlets is a RESTful web framework for Java, http://www.restlet.org/



Step 1: Pick a Taxonomy

Our system supports a variety of naming schemes for organizing data
Select Taxonomy:

© GeoSensor Web Lab 2012

Step 2: Pick a Virtual Layer

Step 3: Browse the Data

Using the map above, browse the available sensor data!

Current Layer Loaded:
urn:ogc:def:property:ucalgary:gsw:soil_temperature

Figure 5.3: VirtualSOS, a Web Based Application of Grouping Similar Sensor Data Layers

Chapter 6

Conclusions

We start this thesis by comparing the rise of the World Wide Web (WWW) in the early 1990s to today's rising Sensor Web. The Sensor Web runs on top of the Internet through the use of open standards, but even with the use of open standards for sharing information, there are still interoperability issues that arise. Many SDIs rely on accessing data from multiple sources and integrating them seamlessly into a single, logical presentation for the user. Our focus is on the task of grouping semantically similar sensor data layers. This will increase the usability of SDIs by saving users the time of manually sorting through data layers.

However, there are many problems associated with this task. The sheer number of unique sensors will necessitate an automatic approach, as manual categorization will not be feasible with the rising number of sensors. As well, the heterogeneous naming of sensor data layers makes it difficult to perform exact string matching. One well researched solution to this problem has been the semantic catalogue. These catalogues automatically groups together heterogeneous data sources. However, this approach requires the creation and maintenance of ontologies, which is a very time intensive process. As well, real world data providers often do not provide ontologies with their data, which makes it extremely difficult to use that data in a semantic catalogue.

The automatic grouping of sensor data layers presents two primary challenges in the form of differences between names. These are syntactic and semantic differences. Syntactic differences are resolved to some extent by open standards, but the same name can be represented using different characters. The best example of syntactic differences is the use of uppercase and lowercase to represent the same name. Semantic differences are more difficult to resolve, and refer to two different names to represent the same real world concept. Our thesis contributes a great deal to the GIS community. We present here an evaluation of syntactic and semantic string matching algorithms for the purposes of automatically grouping similar sensor data layers. We investigate the SOS standard in detail, and apply clustering and classification to a novel data set, GIS data layers.

Our methodology is a solid bottom-up approach. First data is collected from different OGC SOS services. Then it is divided into atomic data layers known as Property Layers (PLs). The text from PLs that convey information about the phenomenon the PLs measure is processed via normalization and tokenization. Next, WordNet is introduced as a lexical database to create word pair similarity scores. Many dissimilarity functions are introduced, based on approximate string matching. Using these dissimilarity functions, we perform PL-PL mapping, PL clustering, and class-PL mapping.

We present an evaluation of how these dissimilarity functions performed in grouping similar sensor data layers. Overall, we see comparable results using edit-based and set-based dissimilarity functions. The semantic dissimilarity function did not perform as expected, and often did not perform very well. The best semantic dissimilarity function was one that only considered very direct and simple relationships between tokens.

6.0.1 Future Work

The research presented here can be extended so more meaningful groups of data layers can be created. One important aspect will be to apply this methodological framework to another data set. Much of this research is designed specifically for the SOS standard, but the methodology is fundamental enough that it could be applied to other geospatial data standards.

It is important to investigate optimization techniques, particularly for clustering data layers. As sensor data layers increase, it may not be possible to cluster extremely large data sets.

Perhaps the most important future work would be to include data providers' ontolgoical data in this methodology. Some data providers do provide ontologies or use ontologies to manage their information, and it would be beneficial to include their ontologies when available. This can be done by using the provided ontologies to assist in defining word pair similarity scores, or even more robust measures of semantic information. This is necessary because WordNet cannot capture contextual information that is relevant to every data source.

Bibliography

[1] K. A. Delin and S. P. Jackson, “The sensor web: A new instrument concept,” 2001.

[2] S. Liang, A. Croitoru, and C. V. Tao, “A distributed geospatial infrastructure for sensor web,” *Computers & Geosciences*, vol. 31, pp. 221–231, Mar. 2005.

[3] P. Hartwell, “How a physically aware internet will change the world.” http://mashable.com/2010/10/13/sensors-internet/, 2010.

[4] I. Bramson, “Mother earth gets a central nervous system: 1 trillion sensors | web simple.” http://websimpletools.com/2010/09/mother-earth-gets-a-central-nervous-system-1-trillion-sensors/, 2010.

[5] S. Cox, “Geographic information: Observations and measurements,” Nov. 2010.

[6] J. Nogueras-Iso, J. Zarazaga-Soria, and P. Muro-Medrano, *Geographic Information Metadata for Spatial Data Infrastructures.* Springer, 2005.

[7] D. J. Coleman and D. D. Nebert, “Building a north american spatial data infrastructure,” *Cartography and Geographic Information Science*, vol. 25, pp. 151–160, July 1998.

[8] B. Knoechel, C.-Y. Huang, and S. Liang, “Design and implementation of a system for the improved searching and accessing of real-world SOS services,” (Banff, Canada), Oct. 2011.

[9] S. Jirka, A. Broring, and T. Foerster, “Handling the semantics of sensor observables within SWE discovery solutions,” in *Collaborative Technologies and Systems (CTS), 2010 International Symposium on*, p. 322–329, 2010.

[10] A. Bröring, P. Maué, K. Janowicz, D. Nüst, and C. Malewski, “Semantically-enabled sensor plug & play for the sensor web,” *Sensors*, vol. 11, pp. 7568–7605, Aug. 2011.

[11] R. Rezel and S. Liang, “A folksonomy-based recommendation system for the sensor web,” *Web and Wireless Geographical Information Systems*, p. 64–77, 2011.

[12] C. Manning, P. Raghavan, and H. Schütze, “Introduction to information retrieval.” http://nlp.stanford.edu/IR-book/html/htmledition/irbook.html, 2008.

[13] M. Hadjieleftheriou, “Approximate string processing,” *Foundations and Trends in Databases*, vol. 2, no. 4, pp. 267–402, 2009.

[14] W. E. Winkler, “Overview of record linkage and current research directions,” 2006.

[15] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, “A comparison of string distance metrics for name-matching tasks,” in *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, p. 73–78, 2003.

[16] E. Rahm and P. A. Bernstein, “A survey of approaches to automatic schema matching,” *The VLDB Journal*, vol. 10, pp. 334–350, Dec. 2001.

[17] I. Cruz, F. Antonelli, and C. Stroe, “Efficient selection of mappings and automatic quality-driven combination of matching methods,” in *ISWC International Workshop on Ontology Matching. CEUR-WS*, 2009.

[18] P.-N. Tan, S. Michael, and K. Vipin, *Introduction to Data Mining.* Boston: Pearson Education, Inc., 2006.

[19] S. Shehata, F. Karray, and M. Kamel, “An efficient concept-based mining model for enhancing text clustering,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, 2010.

[20] J.-W. v. Dam, D. Vandic, F. Hogenboom, and F. Frasincar, “Searching and browsing tag spaces using the semantic tag clustering search framework,” pp. 436–439, IEEE, Sept. 2010.

[21] J. Gemmell, A. Shepitsen, B. Mobasher, and R. Burke, “Personalizing navigation in folksonomies using hierarchical tag clustering,” *Data Warehousing and Knowledge Discovery*, p. 196–205, 2008.

[22] Y. Bishr, “Overcoming the semantic and other barriers to GIS interoperability,” *International Journal of Geographical Information Science*, vol. 12, pp. 299–314, June 1998.

[23] W. Kuhn, “Semantic reference systems,” *International Journal of Geographical Information Science*, vol. 17, pp. 405–409, June 2003.

[24] T. Pedersen, S. Patwardhan, and J. Michelizzi, “WordNet:: similarity: measuring the relatedness of concepts,” in *Demonstration Papers at HLT-NAACL 2004*, p. 38–41, 2004.

[25] T. R. Gruber *et al.*, “A translation approach to portable ontology specifications,” *Knowledge acquisition*, vol. 5, no. 2, p. 199–220, 1993.

[26] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins, “What are ontologies, and why do we need them?,” *Intelligent Systems and Their Applications, IEEE*, vol. 14, no. 1, p. 20–26, 1999.

[27] S. Staab and R. Studer, eds., *Handbook on Ontologies.* International Handbooks on Information Systems, 2009.

[28] N. Guarino and P. Giaretta, “Ontologies and knowledge bases towards a terminological clarification,” pp. 25–32, 1995.

[29] X. Su and L. Ilebrekke, “A comparative study of ontology languages and tools,” in *Advanced Information Systems Engineering*, p. 761–765, 2006.

[30] N. Lemos, *An introduction to the theory of knowledge.* Cambridge UK ;;New York: Cambridge University Press, 2007.

[31] M. Worboys and M. Duckham, *GIS A Computing Perspective*. Crc Press, 2 ed., 2004.

[32] A. Pease, I. Niles, and J. Li, "The suggested upper merged ontology: A large ontology for the semantic web and its applications," in *Working Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web*, vol. 28, 2002.

[33] H. Zhu and S. Madnick, "A lightweight ontology approach to scalable interoperability," in *VLDB Workshop on Ontologies-based techniques or DataBases and Information Systems (ODBIS'06)*, 2006.

[34] D. Gasevic, D. Djuric, and V. Devedzic, *Model driven architecture and ontology development*. Springer-Verlag, 2006.

[35] O. Corcho and A. Gómez-Pérez, "A roadmap to ontology specification languages," *Knowledge Engineering and Knowledge Management Methods, Models, and Tools*, p. 80–96, 2000.

[36] A. Gangemi, R. Navigli, and P. Velardi, "The OntoWordNet project: extension and axiomatization of conceptual relations in WordNet," *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, p. 820–838, 2003.

[37] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American Magazine*, May 2001.

[38] T. Berners-Lee, "Linked data - design issues." http://www.w3.org/DesignIssues/LinkedData.html, 2006.

[39] F. Manola and E. Miller, "RDF primer." http://www.w3.org/TR/rdf-primer/, 2004.

[40] J. Pan, "Resource description framework," *Handbook on Ontologies*, p. 71–90, 2009.

[41] G. Antoniou and F. Harmelen, "Web ontology language: Owl," *Handbook on ontologies*, p. 91–110, 2009.

[42] "SPARQL query language for RDF." http://www.w3.org/TR/rdf-sparql-query/, 2008.

[43] L. for Applied Ontology, "DOLCE : a descriptive ontology for linguistic and cognitive engineering." http://www.loa-cnr.it/DOLCE.html, 2011.

[44] A. Pease, "Suggested upper merged ontology (SUMO)." http://www.ontologyportal.org/, 2011.

[45] H. Stenzhorn, "Basic formal ontology (BFO) | home." http://www.ifomis.org/bfo/, 2011.

[46] S. K. Technologies, "PROTON ontology home." http://proton.semanticweb.org/, 2011.

[47] C. Henson, J. Pschorr, A. Sheth, and K. Thirunarayan, "SemSOS: semantic sensor observation service," in *Collaborative Technologies and Systems, 2009. CTS'09. International Symposium on*, p. 44–53, 2009.

[48] K. Janowicz, M. Wilkes, and M. Lutz, "Similarity-based information retrieval and its role within spatial data infrastructures," *Geographic Information Science*, p. 151–167, 2008.

[49] M. Lutz, J. Sprado, E. Klien, C. Schubert, and I. Christ, "Overcoming semantic heterogeneity in spatial data infrastructures," *Computers & Geosciences*, vol. 35, pp. 739–752, Apr. 2009.

[50] A. Na and M. Priest, "Sensor observation service," Oct. 2007.

[51] M. Botts, "OpenGIS sensor model language (SensorML) implementation specification," July 2007.

[52] S. Chen and S. Liang, "A hybrid peer-to-peer architecture for global geospatial web service discovery," (Canada), 2011.

[53] A. Budanitsky and G. Hirst, "Evaluating wordnet-based measures of lexical semantic relatedness," *Computational Linguistics*, vol. 32, no. 1, p. 13–47, 2006.

[54] P. University, "About WordNet - WordNet - about WordNet." http://wordnet.princeton.edu/, 2011.

[55] F. Lin and K. Sandkuhl, "A survey of exploiting wordnet in ontology matching," in *Artificial Intelligence in Theory and Practice II* (M. Bramer, ed.), vol. 276 of *IFIP International Federation for Information Processing*, pp. 341–350, Springer Boston, 2008.

[56] M. Warin, H. Oxhammar, and M. Volk, "Enriching an ontology with wordnet based on similarity measures," in *MEANING-2005 Workshop*, 2005.

[57] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," *Arxiv preprint cmp-lg/9511007*, 1995.

[58] D. Lin, "An information-theoretic definition of similarity," in *Proceedings of the 15th international conference on Machine Learning*, vol. 1, p. 296–304, 1998.

[59] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," *Arxiv preprint cmp-lg/9709008*, 1997.

[60] G. Hirst and E. St-Onge, "Lexical chains as representations of context for the detection and correction of malapropisms," *WordNet: An electronic lexical database*, pp. 305–332, 1995.

[61] S. Banerjee and T. Pedersen, "Extended gloss overlaps as a measure of semantic relatedness," in *International Joint Conference on Artificial Intelligence*, vol. 18, p. 805–810, 2003.

[62] L. Kaufman and P. Rousseeuw, "Clustering by means of medoids," in *Statistical Data Analysis Based on the L1 Norm* (Y. Dodge, ed.), pp. 405–416, Amsterdam: North Holland, 1987.

[63] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” pp. 226–231, AAAI Press, 1996.

Appendix A

Sample GetCapabilities File

```
<sos:Capabilities xmlns:sos=”http://www.opengis.net/sos/1.0”
xmlns:gml=”http://www.opengis.net/gml”
xmlns:xlink=”http://www.w3.org/1999/xlink”
xmlns:ows=”http://www.opengis.net/ows/1.1”
xmlns:swe=”http://www.opengis.net/swe/1.0.1”
xmlns:om=”http://www.opengis.net/om/1.0”
xmlns:xsi=”http://www.w3.org/2001/XMLSchema–instance”
version=”1.0.0”
xsi:schemaLocation=”http://www.opengis.net/sos/1.0
http://schemas.opengis.net/sos/1.0.0/sosAll.xsd”>
<ows:ServiceIdentification>
<ows:ServiceType>OGC:SOS</ows:ServiceType>
<ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
<ows:Title>Buoy Data from NDBC</ows:Title>
</ows:ServiceIdentification>
<ows:ServiceProvider>
<ows:ProviderName>NDBC</ows:ProviderName>
<ows:ProviderSite xlink:href=”http://geocens.ca”/>
<ows:ServiceContact>
<ows:IndividualName>GSW</ows:IndividualName>
<ows:PositionName>lab admin</ows:PositionName>
<ows:ContactInfo>
```

```
<ows:Phone>
<ows:Voice>403-220-8038</ows:Voice>
</ows:Phone>
<ows:Address>
<ows:DeliveryPoint>Calgary</ows:DeliveryPoint>
<ows:City>Calgary</ows:City>
<ows:AdministrativeArea>NW</ows:AdministrativeArea>
<ows:PostalCode>T2N 1N4</ows:PostalCode>
<ows:Country>Canada</ows:Country>
<ows:ElectronicMailAddress>gsw@ucalgary.ca
</ows:ElectronicMailAddress>
</ows:Address>
</ows:ContactInfo>
<ows:Role>SOS Developer</ows:Role>
</ows:ServiceContact>
</ows:ServiceProvider>
<ows:OperationsMetadata>
<ows:Operation name=”GetCapabilities”>
<ows:DCP>
<ows:HTTP>
<ows:Get xlink:href=”http://136.159.121.217:8171/sos?”/>
<ows:Post xlink:href=”http://136.159.121.217:8171/sos”/>
</ows:HTTP>
</ows:DCP>
<ows:Parameter name=”AcceptVersions”>
<ows:AllowedValues>
```

```
<ows:Value>1.0.0</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
</ows:Operation>
<ows:Operation name=”GetObservation”>
<ows:DCP>
<ows:HTTP>
<ows:Post xlink:href=”http://136.159.121.217:8171/sos”/>
</ows:HTTP>
</ows:DCP>
<ows:Parameter name=”version”>
<ows:AllowedValues>
<ows:Value>1.0.0</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
<ows:Parameter name=”service”>
<ows:AllowedValues>
<ows:Value>SOS</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
<ows:Parameter name=”srsName”>
<ows:AllowedValues>
<ows:Value>EPSG4326</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
<ows:Parameter name=”offering”>
```

```
<ows:AllowedValues>
<ows:Value>Wind</ows:Value>
<ows:Value>Temperature</ows:Value>
<ows:Value>Pressure</ows:Value>
<ows:Value>Wave</ows:Value>
<ows:Value>Visibility</ows:Value>
<ows:Value>Tide</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
<ows:Parameter name=”eventTime”/>
<ows:Parameter name=”observedProperty”>
<ows:AllowedValues>
<ows:Value>Wind Direction</ows:Value>
<ows:Value>Water Temperature</ows:Value>
<ows:Value>Dew Point</ows:Value>
<ows:Value>Air Temperature</ows:Value>
<ows:Value>Atmospheric Pressure</ows:Value>
<ows:Value>Wind Speed</ows:Value>
<ows:Value>Significant Wave Height</ows:Value>
<ows:Value>Mean Wave Direction</ows:Value>
<ows:Value>Dominant Wave Period</ows:Value>
<ows:Value>Pressure Tendency</ows:Value>
<ows:Value>Wind Gust</ows:Value>
<ows:Value>Visibility</ows:Value>
<ows:Value>Tide</ows:Value>
<ows:Value>Average Wave Period</ows:Value>
```

```
</ows:AllowedValues>
</ows:Parameter>
<ows:Parameter name=”procedure”>
<ows:AllowedValues>
<ows:Value>http://136.159.121.217:8171/sos/procedures</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
<ows:Parameter name=”featureOfInterest”>
<ows:AllowedValues>
<ows:Value>http://136.159.121.217:8171/sos/fois</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
<ows:Parameter name=”resultModel”>
<ows:AllowedValues>
<ows:Value>om:Observation</ows:Value>
<ows:Value>om:Measurement</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
<ows:Parameter name=”responseFormat”>
<ows:AllowedValues>
<ows:Value>text/xml;subtype=”om/1.0.0”</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
</ows:Operation>
<ows:Operation name=”DescribeSensor”>
<ows:DCP>
```

```
<ows:HTTP>
<ows:Post xlink:href=”http://136.159.121.217:8171/sos”/>
</ows:HTTP>
</ows:DCP>
<ows:Parameter name=”version”>
<ows:AllowedValues>
<ows:Value>1.0.0</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
<ows:Parameter name=”outputFormat”>
<ows:AllowedValues>
<ows:Value>text/xml;subtype=”sensorML/1.0.1”</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
<ows:Parameter name=”procedure”>
<ows:AllowedValues>
<ows:Value>http://136.159.121.217:8171/sos/procedures</ows:Value>
</ows:AllowedValues>
</ows:Parameter>
</ows:Operation>
</ows:OperationsMetadata>
<sos:Contents>
<sos:ObservationOfferingList>
<sos:ObservationOffering gml:id=”Wind”>
<gml:name>Wind</gml:name>
<gml:srsName>urn:ogc:crs:epsg:4326</gml:srsName>
```

```
<gml:boundedBy>
<gml:Envelope srsName=”urn:ogc:crs:epsg:4326”>
<gml:lowerCorner>−19.713 −177.75</gml:lowerCorner>
<gml:upperCorner>70.4 175.27</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
<sos:time>
<gml:TimePeriod>
<gml:beginPosition>2011−08−23T18:00:00Z</gml:beginPosition>
<gml:endPosition>2012−02−24T23:59:59Z</gml:endPosition>
</gml:TimePeriod>
</sos:time>
<sos:procedure xlink:href=
”http://136.159.121.217:8171/sos/offerings/Wind/procedures”/>
<sos:observedProperty xlink:href=
”urn:ogc:def:property:noaa:ndbc:Wind␣Direction”/>
<sos:observedProperty xlink:href=
”urn:ogc:def:property:noaa:ndbc:Wind␣Speed”/>
<sos:observedProperty xlink:href=
”urn:ogc:def:property:noaa:ndbc:Wind␣Gust”/>
<sos:featureOfInterest xlink:href=
”http://136.159.121.217:8171/sos/offerings/Wind/fois”/>
<sos:responseFormat>text/xml;subtype=”om/1.0.0”
</sos:responseFormat>
<sos:resultModel>Measurement</sos:resultModel>
<sos:resultModel>Observation</sos:resultModel>
```

```
</sos:ObservationOffering>
<sos:ObservationOffering gml:id=”Temperature”>
<gml:name>Temperature</gml:name>
<gml:srsName>urn:ogc:crs:epsg:4326</gml:srsName>
<gml:boundedBy>
<gml:Envelope srsName=”urn:ogc:crs:epsg:4326”>
<gml:lowerCorner>−19.713 −178.343</gml:lowerCorner>
<gml:upperCorner>70.4 175.27</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
<sos:time>
<gml:TimePeriod>
<gml:beginPosition>2011−08−24T00:00:00Z</gml:beginPosition>
<gml:endPosition>2012−02−24T23:59:59Z</gml:endPosition>
</gml:TimePeriod>
</sos:time>
<sos:procedure xlink:href=
”http://136.159.121.217:8171/sos/offerings/Temperature/procedures”/>
<sos:observedProperty xlink:href=
”urn:ogc:def:property:noaa:ndbc:Water␣Temperature”/>
<sos:observedProperty xlink:href=
”urn:ogc:def:property:noaa:ndbc:Dew␣Point”/>
<sos:observedProperty xlink:href=
”urn:ogc:def:property:noaa:ndbc:Air␣Temperature”/>
<sos:featureOfInterest xlink:href=
”http://136.159.121.217:8171/sos/offerings/Temperature/fois”/>
```

```
<sos:responseFormat>text/xml;subtype=”om/1.0.0”
</sos:responseFormat>
<sos:resultModel>Measurement</sos:resultModel>
<sos:resultModel>Observation</sos:resultModel>
</sos:ObservationOffering>
<sos:ObservationOffering gml:id=”Pressure”>
<gml:name>Pressure</gml:name>
<gml:srsName>urn:ogc:crs:epsg:4326</gml:srsName>
<gml:boundedBy>
<gml:Envelope srsName=”urn:ogc:crs:epsg:4326”>
<gml:lowerCorner>-19.713 -177.75</gml:lowerCorner>
<gml:upperCorner>70.4 175.27</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
<sos:time>
<gml:TimePeriod>
<gml:beginPosition>2011-08-24T00:00:00Z</gml:beginPosition>
<gml:endPosition>2012-02-24T23:59:59Z</gml:endPosition>
</gml:TimePeriod>
</sos:time>
<sos:procedure xlink:href=
”http://136.159.121.217:8171/sos/offerings/Pressure/procedures”/>
<sos:observedProperty xlink:href=
”urn:ogc:def:property:noaa:ndbc:Atmospheric␣Pressure”/>
<sos:observedProperty xlink:href=
”urn:ogc:def:property:noaa:ndbc:Pressure␣Tendency”/>
```

```
<sos:featureOfInterest xlink:href=
”http://136.159.121.217:8171/sos/offerings/Pressure/fois”/>
<sos:responseFormat>text/xml;subtype=”om/1.0.0”
</sos:responseFormat>
<sos:resultModel>Measurement</sos:resultModel>
<sos:resultModel>Observation</sos:resultModel>
</sos:ObservationOffering>
<sos:ObservationOffering gml:id=”Wave”>
<gml:name>Wave</gml:name>
<gml:srsName>urn:ogc:crs:epsg:4326</gml:srsName>
<gml:boundedBy>
<gml:Envelope srsName=”urn:ogc:crs:epsg:4326”>
<gml:lowerCorner>−19.691 −177.75</gml:lowerCorner>
<gml:upperCorner>65.698 175.27</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
<sos:time>
<gml:TimePeriod>
<gml:beginPosition>2011−08−24T00:00:00Z</gml:beginPosition>
<gml:endPosition>2012−02−24T23:59:59Z</gml:endPosition>
</gml:TimePeriod>
</sos:time>
<sos:procedure xlink:href=
”http://136.159.121.217:8171/sos/offerings/Wave/procedures”/>
<sos:observedProperty xlink:href=
”urn:ogc:def:property:noaa:ndbc:Significant␣Wave␣Height”/>
```

```
<sos:observedProperty xlink:href=
"urn:ogc:def:property:noaa:ndbc:Mean␣Wave␣Direction"/>
<sos:observedProperty xlink:href=
"urn:ogc:def:property:noaa:ndbc:Dominant␣Wave␣Period"/>
<sos:featureOfInterest xlink:href=
"http://136.159.121.217:8171/sos/offerings/Wave/fois"/>
<sos:responseFormat>text/xml;subtype="om/1.0.0"
</sos:responseFormat>
<sos:resultModel>Measurement</sos:resultModel>
<sos:resultModel>Observation</sos:resultModel>
</sos:ObservationOffering>
<sos:ObservationOffering gml:id="Visibility">
<gml:name>Visibility</gml:name>
<gml:srsName>urn:ogc:crs:epsg:4326</gml:srsName>
<gml:boundedBy>
<gml:Envelope srsName="urn:ogc:crs:epsg:4326">
<gml:lowerCorner>28.867 -92.061</gml:lowerCorner>
<gml:upperCorner>61.4 2.8</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
<sos:time>
<gml:TimePeriod>
<gml:beginPosition>2011-08-24T00:00:00Z</gml:beginPosition>
<gml:endPosition>2012-02-24T23:59:59Z</gml:endPosition>
</gml:TimePeriod>
</sos:time>
```

```
<sos:procedure xlink:href=
”http://136.159.121.217:8171/sos/offerings/Visibility/procedures”/>
<sos:observedProperty xlink:href=
”urn:ogc:def:property:noaa:ndbc:Visibility”/>
<sos:featureOfInterest xlink:href=
”http://136.159.121.217:8171/sos/offerings/Visibility/fois”/>
<sos:responseFormat>text/xml;subtype=”om/1.0.0”
</sos:responseFormat>
<sos:resultModel>Measurement</sos:resultModel>
<sos:resultModel>Observation</sos:resultModel>
</sos:ObservationOffering>
<sos:ObservationOffering gml:id=”Tide”>
<gml:name>Tide</gml:name>
<gml:srsName>urn:ogc:crs:epsg:4326</gml:srsName>
<gml:boundedBy>
<gml:Envelope srsName=”urn:ogc:crs:epsg:4326”>
<gml:lowerCorner>24.627 −97.05</gml:lowerCorner>
<gml:upperCorner>30.06 −80.433</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
<sos:time>
<gml:TimePeriod>
<gml:beginPosition>2011−08−24T00:00:00Z</gml:beginPosition>
<gml:endPosition>2012−02−24T23:59:59Z</gml:endPosition>
</gml:TimePeriod>
</sos:time>
```

```
<sos:procedure xlink:href=
”http://136.159.121.217:8171/sos/offerings/Tide/procedures”/>
<sos:observedProperty xlink:href=
”urn:ogc:def:property:noaa:ndbc:Tide”/>
<sos:featureOfInterest xlink:href=
”http://136.159.121.217:8171/sos/offerings/Tide/fois”/>
<sos:responseFormat>text/xml;subtype=”om/1.0.0”
</sos:responseFormat>
<sos:resultModel>Measurement</sos:resultModel>
<sos:resultModel>Observation</sos:resultModel>
</sos:ObservationOffering>
</sos:ObservationOfferingList>
</sos:Contents>
</sos:Capabilities>
```