# USING DECISION TREES FOR INDUCTIVELY DRIVEN SEMANTIC INTEGRATION AND ONTOLOGY MATCHING

Bart Gajderowicz

BSc, Ryerson University, Toronto, Canada, 2008

A thesis

presented to Ryerson University

in partial fulfilment of the

requirements for the degree of

Master of Science

In the program of

Computer Science

Toronto, Ontario, Canada, 2011

I hereby declare that I am the sole author of this thesis or dissertation.

I authorize Ryerson University to lend this thesis or dissertation to other institutions or individuals for the purpose of scholarly research.

_____

Place, Date

_____

Signature

I further authorize Ryerson University to reproduce this thesis or dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

Place, Date

_____

Signature

# USING DECISION TREES FOR INDUCTIVELY DRIVEN SEMANTIC INTEGRATION AND ONTOLOGY MATCHING

**Bart Gajderowicz**
**MSc, Computer Science, Ryerson University, 2011**

## Abstract

The popularity of ontologies for representing the semantics behind many real-world domains has created a growing pool of ontologies on various topics. While different ontologists, experts, and organizations create the vast majority of ontologies, often for internal use or for use in a narrow context, their domains frequently overlap in a wider context, specifically for complementary domains. To assist in the reuse of ontologies, this thesis proposes a bottom-up technique for creating concept anchors that are used for ontology matching. Anchors are ontology concepts that have been matched to concepts in an external ontology. The matching process is based on inductively derived decision trees rules for an ontology that are compared with rules derived for external ontologies. The matching algorithm is intended to match taxomonies, ontologies which define subsumption relations between concepts, with an associated database used to derive the decision trees. This thesis also introduces several algorithm evolution measures, and presents a set of use cases that demonstrate the strengths and weaknesses of the matching process.

Keywords: ontology matching, machine learning, decision trees, semantic integration, taxonomy.

# Acknowledgments

This thesis is the result of a great deal of time assessing the hard problems which prohibit people and societies from having an open dialogue with one another, and searching for ways in which my technical background can contribute to these problems. I have spent several years researching these topics, and articulating my findings to the people I credit with helping me achieve this goal. To that end, I would like to thank the following individuals that have assisted me in this process.

First I would like to thank my supervisor Alireza Sadeghian for indulging my ideas at the beginning of this journey, and guiding me as I began to explore the world of formal research. His openness and willingness to lead me through the many potential avenues, helped me focus on the areas necessary to achieve my goals.

I would like to thank my co-supervisor Mikhail Soutchanski for helping me articulate my ideas, and for strengthening my understanding of the logical foundations of ontologies that have guided my interests and the development of my research. His endless assistance and guidance have generated many ideas found within the pages of this thesis.

I would also like to thank Michael Grüninger for the invaluable insight into ontologies and research I have gained through my discussions with him.

Finally, I would like to thank my friends and colleagues Hossein Rahnama and Chris Mawson, who spent countless hours with me discussing the technical, social, and theoretical aspects of my research, and the countless others who have shared their ideas with me.

# Dedications

This thesis is dedicated to my family. To my mother and sister whom I owe everything, and to my wife, without whose patience I could not have reached this goal.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1. Introduction

## 1.1 Problem Statement and Motivation

In today's open community, more organizations are willing to share their data in the hopes of improving their processes through collaboration [1]. Often the individual system's intended domain models, such as concepts, relations and attributes[1], are very similar, if not equivalent to other representations within the same or complimentary domain [2][3]. These domain elements can be represented as a database schema or be embedded in the application's program logic. With different technical representations and implementation details, aligning these elements between two systems can be nearly impossible. To separate the implementation details from the domain elements, many organizations create a higher level description of their domains, through representation structures such as UML diagrams, workflows, rules, or taxonomies [3][4], and often a more formal ontology [5]. This thesis adopts the definition of an *ontology* as a shared understanding of a particular domain through conceptualization, and the use of explicit concepts, definitions, and the relationships between those concepts [6]. A definition more relevant to this thesis, and one which will be expanded later on, is that an *ontology* can also be seen as describing a particular domain using role hierarchies, where roles define concepts and properties of those concepts and relationships among concepts [7]. Ontologies can express a variety of entities, concepts, relationships, processes, roles, guidelines, and others [6]. An ontology's expressiveness depends greatly on the language it is represented in. Ontological elements, similar to implementation details, are open to interpretation by the domain expert, and need to meet the technical requirements and constraints faced by the system architect [8].

The way information is translated into ontologies depends heavily on the ontologist who is creating them, the subject matter expert (SME), and when the ontologies were created [9]. It is often the case that two ontologies are equivalent though their underlying concepts and definitions. It may also be the case that ontologies describing the same phenomenon are made up of differently organized definitions that are not equivalent. This may be due to the SME's bias, ontologist's bias, incomplete data, limited number of finite observations or different contexts. Regardless of the reason it is a well documented fact that conceptualizations by different people will often be translated in different ways [9]. The discrepancies may exist in the interpretations, and also in the available data. The alignment of ontologies is a difficult process, and this thesis focuses on reducing the difficulties in aligning ontologies, from here on called "ontology matching" [10]. The focus of this thesis is taxonomic ontologies, represented by sub-class and super-class relationships.

[1] See Glossary on page 112 for the definitions of concepts, relations, and attributes.

## 1.2 Existing Approaches

Previous work that attempts automatic or semi-automatic ontology matching uses various aspects of the ontology itself. In all cases, however, a similarity measure of some kind is required for the information extraction process. Similarities can be structural in nature, especially when the ontology can be represented by a directed labeled graph, or bipartite graphs [11]. In such cases, lexical *anchors*[2] are found which are initially used to match similar concepts between ontologies, and then their structural position relative to other anchors is considered for further matching [12]. One of the simplest structural characteristics is the relative distance between concept nodes in a structure [13][14]. Considering the distance between nodes is called edge-counting, where the term *edge* is adopted from a graph representation of subsumption. For example, considering $Rain \sqsubseteq Precipitation \sqsubseteq Weather \sqsubseteq Event$, there is 1 edge between *Rain* and *Precipitation*, 2 edges between *Rain* and *Weather*, and 3 edges between *Rain* and *Event*. A more in-depth structure analysis of ontologies can be applied using various graph-theoretic methods. Stochastic measures such as expectation maximization algorithms have been utilized for ontology matching [12][15], by presenting the ontologies as graphs, and performing graph matching [16].

Some methods concentrate on purely lexical information, and look for similarities between labels [15][17][18][19] and data values. This lexical information can relate to record identifiers such as identification values (ID's) or uniform resources locators (URI's), as well as labels which are normalized through word similarities and synonyms [17]. Tools such as WordNet [20] are meant to provide similar terms such as synonyms, as well as much richer semantics between words such as hypernyms[3], hopenym[4], holonyms[5] and troponyms[6] to name a few. Some domains, such as Biomedicine which have a well defined vocabulary, are well suited for such lexical mappings [21]. Various methods have been used to enhance the initial anchors, such as neural networks, decision trees, and support vector machines [22]. A set of terminology relationships such as synonyms, word senses, and hypernym groupings, can be rated and grouped together into a structure like the decision tree, and compared to others in order to find a set of similar characteristics [23][24].

[2] An anchor is a concept in the local ontology which was identified as a match with a target concept. An anchor is used to align ontologies, and possibly perform further matches using structural or other matching techniques.
[3] Hypernym: a term with a broader meaning (e.g. Weather is a hypernym of Snow).
[4] Hopenym: a term with a specialized meaning; opposite of hypernym (e.g. Snow is a hopenym of Weather).
[5] Holonym: a term that denotes a whole made up of parts (e.g. Car is a holonym of engine, wheels, etc).
[6] Troponym: A verb version of hopenym, giving a specialized meaning of a verb (e.g. Strolling is a troponym for walking).

As data becomes more complex, this information also becomes a more powerful resource for matching the ontological elements it is meant to represent. This more complex data will also allow machine learning techniques to advance beyond simple analysis of the format or string sequences of data values, as discussed in the previous paragraph, and instead will allow the analysis of both overall statistical characteristics and underlying patterns in the data. With this, however, comes a level of uncertainty, as the results given by machine learning algorithms are predictions, often based on the maximum entropy principle [25]. Though uncertainty and probabilities have been widely studied in the ontology field, Description Logics are a newer area of interest generating research in recent years in this field [26]. Many researchers concentrate on belief measures of uncertainty introduced by ontological definitions [27][28][29][30], with others concentrating on probabilistic measures [31][32][33][34][35]. Once preliminary *anchors* are defined with associated belief measures, similarity measures can be calculated between two concepts in similar ontologies [36][37].

## 1.3 Methodology

To overcome some of the limitations, and extend the efforts of previous work on ontology matching, we propose an algorithm that inductively builds decision trees based on the ontological characteristics of a data set. The focus is to avoid using *lexical* information of labels and vocabularies of the individual ontologies for creating concept *anchors*. Instead, inductively derived information in a bottom-up fashion is used to create and rate a set of matches. Similar to Volz [38], by using the ontology as a guide for the data-mining algorithm, we are able to utilize the semantic relations of the ontology and the statistical information of the data, to create a set of rules which encompass knowledge from both sources. The resulting decision trees represent refined definitions of ontology elements, specifically ground terms, based on data. This builds a set of anchors, through a bottom up approach, which can act as anchors in the matching process. The hypothesis is that despite discrepancies in two separate ontologies based on different needs, and built by different architects, the data itself will show some level of consistency. Data is required to match concepts, and the assumption is that even though interpretations may differ, the underlying occurrences captured in data records will remain somewhat consistent [10][38].

The rules defined by the *local* decision trees are then checked by a reasoner for consistency with a set of rules from a *target* ontology and data set. The actual data gathered about observations[7] may be consistent amongst the instances observed by other ontologists, even when the data does not apply

[7] Observations are actual events that occur in the real world, about which data can be gathered.

directly to their domain. If so, this data may be used to match concepts from one domain to the other at a more detailed level based on a *learned* model from instance data [10]. For example, a particular type of precipitation event may be characterized as *Rain* or a lighter *Drizzle*. While the two terms are labeled differently, the actual rainfall recorded, along with wind speeds, temperature, and other information would be consistent amongst all observers. Once a consistent set of rules is found, a match is created between the two concepts *Rain* and *Drizzle*. As will be expanded on in section 2.3, this thesis utilizes OWL 2 to represent the derived rules. Due to OWL 2's adoption by W3C as a web ontology standard, it may increase the adoption of these rules by other external systems. Also, if an ontology is already represented in an OWL 2 compatible ontology, the derived rules could act as extensions to existing OWL 2 concepts.

## 1.4 Objectives

The objective of this thesis is to describe the ontology matching process, which is made up of the following modules. (1) First, a local database is enhanced semantically by associating individual records with the single or multiple ontological concepts they represent. This can be done automatically if records already have a "category" column that can be matched up with a concept in the ontology. (2) A reasoner is used to create a sets of records that are grouped under a particular parent concept. The reasoner resolves super-class and sub-class role hierarchies. (3) These records are then classified with a decision tree algorithm. (4) The derived trees, and the associated rules are then converted into OWL 2 axioms for portability and reasoning tasks. (5) The axioms derived from a local and target ontology are used to find overlapping axioms based solely on the data provided by the records themselves (numeric values are generalized by expanding their range and reducing their requirements, while nominal values use WordNet [20] to generalize the terms). Unlike lexical matching algorithms which use tools such as WordNet to generalize labels of concepts or even attributes, as described in the next chapter, this algorithm generalizes data values for the attributes. (6) Finally, the matches are rated based on the accuracy of the original classification and structural similarities dependent on several structural rating schemes. The rated matches are presented to the user for verification. Due to its data driven approach, this algorithm is a bottom-up matching technique, which focuses on creating anchors for ground terms in an ontology, which represent more specific matches compared to their more ambiguous super-class counterparts [39]. If abstract concepts in different ontologies, higher in the ontological hierarchy, share similarities in data, it may be possible to match them as well, as long as the data-driven similarity anchors can be traced up to the abstract concepts through some data attribute.

The applications which accompany this thesis, listed in Appendix 4 on page 86, utilize the WEKA [25] data-mining package to build decision trees. Ontology analysis and reasoning for combining a database and ontologies is performed on ontologies represented in the OWL 2[8] language, using the HermiT [40] reasoner. As will be expanded on in section 2.3, HermiT reasoning services utilized by this thesis, include verifying if a concept is satisfiable given a particular OWL 2 axioms, and deriving the super-classes and sub-classes of a particular concept. HermiT plays a key role in the matching algorithm because it is used to perform the consistency checks between two mappings, based on the created OWL 2 rules, derived from a decision tree. The OWL API [41] is utilized to build OWL 2 syntax, and to express derived rules, build associations, and represent matching criteria for the HermiT reasoner. All development is done using the Ruby[9] language, with jRuby[10] being the interpreter, for integration with the Java[11] based WEKA, OWL API, and HermiT packages.

## 1.5 Thesis Outline and Contributions

This thesis proposes a bottom-up approach to data driven ontology matching. At the beginning of Chapter 2 we provide necessary background knowledge about the technology utilized in this thesis. Sections 2.1 to 2.3 provide background knowledge on the technologies utilized by the thesis. Section 2.4 describes key areas of research in the ontology matching field. The most effective matching techniques are described, mainly lexical and structural characteristics of ontologies. The utilization of machine learning algorithms is also discussed.

Chapter 3 describes the methodology of this thesis. Section 3.1 describes our ontology matching approach, and includes definitions of key concepts. Section 3.2 describes the semantics of ontologies, and describes the first contribution made, which is how machine learning techniques are utilized to represent those semantics. Section 3.3 describes the process of associating an ontology with database records it is meant to represent. Section 3.4 briefly describes how attributes are matched, and how this process can be implemented. Sections 3.5 to 3.8 describe the second key contributions of the thesis, mainly the rule creation and matching process. These sections include detailed definitions needed to implement the algorithm. Sections 3.9 to 3.12 describe the third key contribution of this thesis, mainly the match rating algorithms. It also demonstrates how these are presented to the user for verification purposes.

[8] OWL 2 is a web ontology language, and is described in greater detail in section 2.3.
[9] Ruby 1.8.7 (http://www.ruby-lang.org/en/)
[10] jRuby 1.4.0 (http://jruby.org/)
[11] Java 1.6 (http://java.sun.com/)

A detailed analysis of the algorithms and the results of the tests are described in Chapter 4, including ontologies used for the tests, and the databases associated with the test ontologies. These last sections also give the final contribution of the thesis, which is the extension of existing ontology matching evaluation measures, proposed specifically for evaluating our algorithm. Sections 4.1 to 4.3 define the evaluation measures used in the Results and Analysis chapter to evaluated the matching algorithm. Section 4.4 lists the tests themselves, and gives motivation for the evaluation measures applied. The use cases are described, with an explanation of the test scenario they cover. In section 4.5 we describe the results of the evaluation tests and in section 4.6 we describe the real-life test results. Both sections are accompanied by detailed analysis and derived conclusions. Based on the analysis in the previous sections, section 4.7 describes the limitations of the algorithm, and considerations that must be taken for the process to produce successful matches.

Chapter 5 serves as the conclusion, summarizing the algorithm and analysis of the results, and lists the key contributions made. Finally, the direction for future work is described, with proposed extensions and improvements to the algorithm. We also suggest possible applications of the produced granules, rules and data-points in OWL 2, outside of the ontology matching application.

The appendices provide: the parameters used for the tests, the complete results of the evaluation measures collected; sample match outputs as presented to the user for verification; OWL 2 representations of the axioms utilized by the algorithm; a list of use cases and benchmark concept matches; and a link to the source code written for the implementation of the algorithm and test execution. Finally the last sections provide the Bibliography and a Glossary of the terms, symbols, acronyms and variable naming conventions used throughout the thesis.

# Chapter 2. Background and Related Work

Most ontology matching techniques use a variety of similarity measures. This chapter gives background knowledge and lists related work, which deals with ontology matching in general, data driven ontology matching, and incorporates decision trees with ontology matching.

## 2.1 Ontology Matching

Ontology matching is the alignment between entities of different ontologies [10]. In contrast to similar work in the literature, ontology matching is the alignment of entities in one ontology, to at most one entity in another. Several issues have been identified as obstacles in manual ontology matching [8][10][42], specifically inconsistency, incompleteness and redundancy. This results in incorrectly defined relationships, missing information, or simply human error. Upper ontologies[12] such as DOLCE [43], OpenCYC [44], SUMO [45], and SWEET [46] have been used to serve as a place for defining general concepts, heavily based on natural language and common sense. Such concepts can then be used to define specific concepts in domain ontologies using general terms. This type of mapping to common general terms has been envisioned as helping in matching one ontology to another, promoting reusability of data, assisting in automated inference as well as natural language processing [10]. Manual ontology creation and mapping has been conducted by ontologists and has been based on their experiences in a particular context. Domain specific ontologies, or simply domain ontologies, are created by subject matter experts and ontology engineers. The drawback of the manual matching process is that the creation and mapping of ontologies is time consuming and error prone [8][27].

An emerging field in ontology analysis is automated extraction of subjects and concepts from text [47] in order to create ontologies as well as automated ontology merging [48]. Research in this field has focused on semi-automated algorithms, as well as tools that assist with the manual process. Cognitive support to humans has been very beneficial in helping ontologists build ontologies [49], this is especially important since human experts in the same domain, and even the same expert at different points in time, create different models for the same phenomenon [9]. The trend in streamlining the mapping processes was dominated by structural mappings in early 2000's, with a shift in focus to semantic mappings in 2007 [49]. The main difference between these trends is that structural mappings are those which only relate schemas to each other, whereas semantic mappings

[12] Upper ontologies are typically large ontologies that define "common sense" definitions in a variety of domains.

also consider data based information, such as how terms are related to each other. This has greatly improved the mapping process, by allowing a system to reason with the relationships of individual terms and concepts. Systems such as OntoClean [50] consider lexical information and metadata of ontologies in an attempt to measure an ontology's validity. OntoClean was used to extract a "backbone" of key terms, analyze their position in the hierarchy, and reorganize them into a more logical structure. An ontology's validity was measured by comparing the success of a query against the system before and after it was processed by OntoClean.

Various techniques have been identified by Euzenat and Shvaiko [10] for automated and semi-automated matching techniques. Each technique depends on the type of data the matching algorithm works with. The first type of techniques are *element-level* techniques, which include algorithms based on strings, languages, constraints being applied to entities (such as types, cardinality, keys) and linguistic resources such as lexicons or related thesauri terms. The second type is *structural-level* techniques, which include graph-based techniques that interpret ontologies as labeled graphs (such as database schemas or taxonomies), taxonomy-based techniques that consider only the specialization relation [51], data-analysis and statistical techniques that consider element properties such as maximums, variance, grouping and number of segments [10]. A different set of characterizations could tell us something about the patterns and distribution of data in a particular domain [10][52]. When datasets share a common domain but not the database, it can be beneficial to instead compare the distance between the sets of records within the different databases. In this case, disjoint *extension comparison techniques* can be applied, which can be based on descriptive statistics of a class as a whole, based on element features [10].

In order to apply these characterizations, a data-enhanced ontology can be created which has data associated with individual ontology entities. This will refine the observations made and increase the probability of finding a match from one ontology to another. The information derived from element-level techniques may also increase the degree of consistency in the translation of observations to axioms. The most beneficiary applications of a record-based ontology matching process are ones with domains that possess data sufficiently descriptive to differentiate ontological concepts. Likely candidate domains are scientific research for which experimental results are only as accurate as their underlying data and historical, and archiving systems such as climate prediction and population census surveys. When qualifying collected specimens or observed phenomena, the investigator often relies on a combination of data-driven and theory-driven information [53]. In fields such as geology, qualifying various types of rock depends greatly on the specimens found and the geologist's

knowledge about the region, rock types, and properties that are infrequently observed and theoretically important. Due to personal bias, some theoretical knowledge may be used incorrectly because of an incorrectly qualified location, for example a *lake* instead of a *stream*. Brodaric et al. [53] observed that more consistent and presumably correct qualifications were exhibited using data-driven information, versus theory-based.

## 2.2 Machine Learning

The idea of *learning* can be interpreted in many different ways: is it the act of acquiring knowledge; of becoming aware of our environment based on that knowledge; or the ability to reason based on that knowledge? Finding a universally agreed upon definition is difficult because the philosophical, theoretical, and technical definitions concentrate on various aspects of the meaning. Therefore taking philosophical and theoretical matters aside, this thesis deals with the definition of learning adopted by the machine learning community, mainly any *computational* approach to learning [54]. Because of the data driven nature of this thesis' topic, it specifically concentrates on the definition adopted by the data-mining field, that is, "a technique for finding and describing structural patterns in data for helping to explain that data and make predictions from it" [25]. As will be described in Chapter 2, current research employs many data-mining algorithms for this purpose. Specifically, ontologies can be analyzed, created, and enhanced using machine learning techniques. There are inductive means that deal with the uncertainty of a set of rules, or the probabilistic measures of a particular rule. This thesis employs decision trees as the data-mining algorithm to capture knowledge.

As a data structure, *decision trees* are used to represent the logical structures of classification rules for domain specific empirical data. The basic algorithm selects the property with the highest information gain for a particular class, and creates disjoint subsets based on that property's values. Ordinal attributes are split into two branches on the $<$ and $\geq$ number restriction. For example a *size* property could be split into *large* and *small* classes based on the number of records and their size values. Nominal attributes are treated as categorically disjoint sets, with as many branches as there are values. For example, the taxonomical $\mathcal{R}$ relation "instance of" (France is an instance of Country) would be able to express the ontological relation $x \; \mathcal{R} \; y$ : $[x \in \{\text{Country}\} \wedge y \in \{\text{France, Italy, Spain}\}]$. A decision tree classifying *Country* would be represented with a parent node *Country*, and three sub-nodes, *France*, *Italy*, and *Spain*. These could be further split on an ordinal property such as *population size* ranges, or another nominal property such as *language*. These subsets are smaller in cardinality than the previous subset, but more exact in precision in classifying a concept. The key factor in the

classifying process is the property and value combinations that identify concepts best and make up the classification rules.

If we are to use decision trees for matching purposes, we must ensure the trees are somewhat consistent, despite being created by different people who may be using different data sets. We can assume the algorithm producing the decision tree is the same as well as the process of converting the result into the final ontology language, and accommodating axioms. What we need is to ensure that the randomness of the algorithm in the learning process has minimal impact on the consistency of the resulting trees for the two ontologies being matched. To minimize these inconsistencies, instead of producing a single decision tree to model each ontology's data set, several smaller decision trees are created, and combined using various techniques. The resulting decision tree is an intelligently generalized version, which does not lose any information but rather is easier to compare to a tree created using a different data set with similar information. Each technique generalizes the set of trees by iteratively combining them in various ways.

*Bagging*, which stands for bootstrap aggregation, works by combining all trees produced using the training data, and testing them out on the learning data[13]. This process actually works better when there are many differences in the trees, a phenomenon which can be increased by ensuring the trees are not pruned[14]. Some models not only produce a score, but probabilities as well, averaging those probabilities to enhance the score is called *bagging with costs*.

*Boosting* is a technique that "boosts" the importance of individual records in the data by assigning weights to those records based on whether they were correctly classified by the derived decision tree. By identifying "hard" and "easy" problems, the next iteration produces a classifier that concentrates on the hard problems of the previous step. Over time, the set of generated classifiers complement each other on the testing data set.

*Stacking* is a technique that compares models built with different classification algorithms. Essentially an iterative *metalearner* is a process that learns which classification algorithm is best.

[13] In machine learning, learning data is a subset of data records used by the algorithm primarily for identifying a pattern inherent in the data. Testing data is a separate set of data records used to test the *learned* pattern.

[14] Pruning refers to the removal of a lower section of a tree branch. This results in a reduction of complexity of the decision tree, but also reduces the level of precision offered by the longer branch.

*Option tree* is a technique that produces a decision tree with two types of nodes: the traditional decision node, and a new type called an *option node*. Unlike a decision node that leads to a single branch, an option node leads to all branches simultaneously. Each branch is then scored against the others, with the best one chosen as the predicate for that point of the formation tree. The option tree technique allows for a generalization that may be scored with various methods.

This thesis implements the *bagging with cost* technique, by utilizing WEKA's *MetaCost*[15] class, to standardize the resulting decision trees and remove the affects of randomness in their creation. The *Bagging* technique and *MetaCost* class are WEKA libraries that are utilized in this thesis, and are discussed in greater detail here.

*Bagging*[16], or bootstrap aggregation, is a process of improving the prediction model over some data set $\mathcal{L}$ [55]. Bagging begins by taking the entire training set $\mathcal{L}$, and creating several training sub-sets $\{\mathcal{L}_k\}$ with replacement, with the same record count as $\mathcal{L}$. Although record selection for each $\mathcal{L}_k$ is random, each $\mathcal{L}_k$ has the same underlying distribution as the original data set $\mathcal{L}$. The classification algorithm then builds a single prediction model $\varphi_k$ for each sub-set in $\mathcal{L}_k$. Each model in $\{\varphi_k\}$ is then used to predict all classes in $\mathcal{L}$, and each record in $\mathcal{L}$ is assigned a vote for each $\varphi_k$. This vote identifies how well each record was classified by each model $\varphi_k$, and is used to score how well each $\varphi_k$ classifies each class in $\mathcal{L}$. See the weka.classifiers.meta.Bagging Java model in WEKA [25] for details of the implementation, and calculation of the vote value. The *MetaCost* class utilizes the calculated votes to create a new dataset $\mathcal{L}_w$. $\mathcal{L}_w$ is a copy of $\mathcal{L}$ where the class of each record is weighed based on the best average vote, from each prediction model in $\{\varphi_k\}$. Finally, the new $\mathcal{L}_w$ is used to create the final decision tree, used by this thesis.

The rules derived by the algorithm are meant to be exchanged by different systems. As a result, a language must be chosen that represents the rules with sufficient expressivity and a wide acceptance by the intended communities. Traditional rule based systems utilize languages and reasoners intended for closed homogenous environments. However systems that work in an open heterogeneous environment must have a different language, which meets a new set of requirements [5].

[15] The *bagging with costs* method utilized in this thesis is the *weka.classifiers.meta.MetaCost* class, which is part of the WEKA package.

[16] The main Bagging algorithm is based on Breiman [55], but the detailed description given here is based on the implementation of WEKA's class *weka.classifiers.meta.Bagging*.

## 2.3 Web Ontology Language OWL

The language chosen here to represent derived rules is the Web Ontology Language 2, or OWL 2, which has been adopted as a recommendation for the web ontology standard by the World Wide Web Consortium (W3C) [56]. Its predecessor OWL was recommended in 2004 to be used as the language to represent ontologies on the semantic web, with the next challenge being extending these ontologies with rules [5].

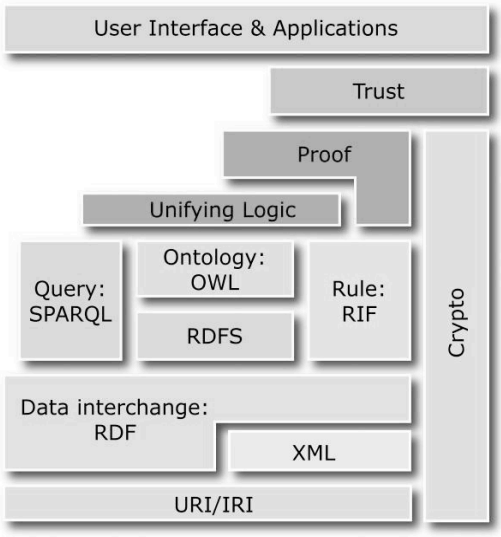



**Figure 1. Latest form of the Semantic Web stack diagram (W3C Semantic Web Activity 2010).**

Figure 1 shows OWL ontologies are parallel with Rule Interchange Format (RIF) rules. RIF has been proposed specifically as the standard for sharing rules over the semantic web[17]. The W3C has published a detailed procedure for representing OWL 2 axioms in RIF[18]. OWL 2 was chosen as the actual representation language in this thesis for two key reasons. The first is the availability of an OWL 2 reasoner called HermiT [40]. No such reasoner currently exists for RIF. The reasoning services provided by HermiT, and utilized by this thesis, include verifying if a concept is satisfiable given particular axioms, and deriving the super-classes and sub-classes of a particular concept. Ultimately, the rules built by the algorithm presented here can be easily converted from RIF to OWL 2 for reasoning purposes, and back to RIF again for sharing. The second key reason is that by representing the rules as OWL 2 axioms, they can be directly embedded in the original ontology, if it

[17] http://www.w3.org/TR/2009/WD-rif-overview-20091001/
[18] http://www.w3.org/TR/2009/WD-rif-owl-rl-20091001/

was written in an OWL 2 comparable language. This would in fact extend the existing ontology and its concepts through a method called Ripple Down Rule (RDR)[19] [57].

**Table 1. Basic Description Logic $\mathcal{AL}$ (Attribute Language)**

| Syntax | Description |
|---|---|
| A, B | Atomic concept |
| C, D | Concept descriptions |
| R | Atomic Role |
| ⊤ | Universal concept |
| ⊥ | Bottom concept |
| ¬A | Atomic negation |
| C ⊓ D | Concept intersection |
| ∀ R.⊥ | Value restriction |
| ∃ R.⊤ | Limited existential quantification |

**Table 2. Family of Description Logics and extensions**

| Name | Characteristic |
|---|---|
| $\mathcal{AL}$ | Basic Description Logic (Attribute Language) |
| $\mathcal{ALU}$ | $\mathcal{AL}$ + unions |
| $\mathcal{ALE}$ | $\mathcal{AL}$ + full existential quantification |
| $\mathcal{ALN}$ | $\mathcal{AL}$ + number restrictions |
| $\mathcal{ALF}$ | $\mathcal{AL}$ + non-atomic negation |
| $\mathcal{ALC}$ | $\mathcal{AL}$ + union ($\mathcal{U}$) + existential quantification ($\mathcal{E}$) + complement ($C$) |
| $S$ | $\mathcal{ALC}$ + transitively closed primitive roles |
| $\mathcal{SH}$ | $S$ + hierarchy of roles |
| $\mathcal{SHI}$ | $\mathcal{SH}$ + inverse roles |
| $\mathcal{SHIQ}$ | $\mathcal{SHI}$ + generalized cardinality restrictions |
| $\mathcal{SHIN}$ | $\mathcal{SHIQ}$ limited to unqualified number restrictions |
| $\mathcal{SHO}$ | $\mathcal{SH}$ + singleton classes |
| $\mathcal{SHOQ}$ | $\mathcal{SHO}$ + qualified number restrictions |
| $\mathcal{SHIF}$ | $\mathcal{SHIN}$ limited to functional number restrictions[20] |
| $\mathcal{SHOIN}$ | $\mathcal{SHOQ}$ + inverse roles ($\mathcal{I}$) + unqualified number restrictions ($\mathcal{N}$) |
| $\mathcal{SROIQ}$ | $\mathcal{SHOIN}$ + complex role inclusion + reflexive roles + transitive roles + irreflexive roles + disjoint roles + universal role + new construct ∃R.Self |

[19] See the Glossary for a definition of Ripple Down Rule.

[20] Functional number restrictions represent a literal that uniquely identifies an individual, such as a Social Insurance Number or Student ID.

OWL 2 is based on Description Logics, which provide constructors to build complex concepts and roles out of atomic ones [58], with various extensions derived to handle different types of constructs [27][58]. The inference capability of Description Logics is provided by the same mechanism used by humans as well as intelligent information processing systems, mainly the classification of concepts and individuals [59]. In this sense, classification of concepts means creating sub-class and super-class relationships between concepts in a hierarchical fashion. In Description Logics these hierarchical relationships are called *subsumptions*, and are identified by the symbol $\sqsubseteq$. The semantics of subsumption are the same as sub-set $\subseteq$, but are applied to subsuming different types of elements, such as atomic concepts, atomic roles (relationships), and their complex counterparts. The complexity of these relationships is based on the expressivity of a particular Description Logic, and more specifically, the characteristics of particular Description Logic extension.

Description Logics in their current form have developed from a basic set of properties of the $\mathcal{AL}$ Description Logic in , to several different extensions, as listed in . In recent years, much attention has been placed on the $\mathcal{SH}$ family of extensions, since it provides sufficient expressivity useful for intended application domains with the use of hierarchical concepts such as *Tree* $\sqsubseteq$ *Plant*, and also hierarchical roles such as *hasBranch* is a sub-role of *hasPart*, represented by *hasBranch* $\sqsubseteq$ *hasPart*. Following $\mathcal{SH}$, the $\mathcal{SHOQ}$ extension has added the capability to specify qualified number restrictions, adding the capability to represent concepts such as "Father with at most 2 female children", represented by *Father* $\sqcap \leq 2$ *hasChild.Female*.

The $\mathcal{SHOIN}$ extension has combined singleton classes, inverse roles and unqualified number restrictions. As a result, $\mathcal{SHOIN}$ has the ability to define an inverse role, such as *childOf* is an inverse of *parentOf*. Unqualified number restriction provides the ability to express the fact that being a parent implies having children, without referring to the children concepts themselves. For example, $\mathcal{SHOIN}$ can represent concepts such as "Father with at most 2 children", represented by *Father* $\sqcap \leq 2$ *hasChild*, which does not require the added concept qualification of *Female* required by previous Description Logic extensions. Further, $\mathcal{SHOIN}$ has been used to create the Web Ontology Language (OWL), which has been adopted as the web ontology standard by W3C [27].

Finally, based on input from ontology developers in practice, Horrocks at el. [60] have extended $\mathcal{SHOIN}$ by adding the characteristics listed in  to create the $\mathcal{SROIQ}$ Description Logic. By allowing for more complex roles, $\mathcal{SROIQ}$ can define the disjoint relationship between *partOf* and *hasPart*

roles. Further, we can identify the reflexivity of the role *knows*, where every person *knows* themselves, and the irreflexivity of the role *siblingOf*, where no person is a sibling of themselves. Negated role assertion removes the need to identify two roles such as *likes* and *dislikes*, by defining dislikes simply as ¬*likes*. We can now say that STEVE likes MARY, but MARY doesn't like STEVE by defining *likes*, and writing *likes*(STEVE, MARY) and ¬*likes*(MARY, STEVE).

Complex role inclusion allows for a mixing of roles. For example, the transitive roles *hasPart* and *owns*, can be combined to state that the owner of C also owns every part of C, represented as *owns* ∘ *hasPart* ⊑ *owns*. For example, this can be used to express that if an engine is part of a car, CAR ⊑ ∃hasPart.ENGINE, and a person owns that car, than that person owns that engine as well, represented as ∃owns.CAR ⊑ ∃owns.ENGINE.

Universal roles define an explicit complete relation between every individual. This allows the definition of a single domain specific class that expresses all axioms in an ontology [60]. For example, a single class *EUCountries* can be constructed by the universal role that groups all European Union countries together. The class *EUCountries* can now be inferred by iterating through European Union countries.

The last characteristic that $\mathcal{SROIQ}$ introduces is the concept of *Self*, which applies to every concept in a domain, and is represented by ∃R.*Self*, where R is a type of relation. This allows the definition of a "locally reflexive" role such as "narcissist" as being someone who likes themselves, defined by ∃*likes*.Self.

Every Description Logic is comprised of two components: the $\mathcal{ABox}$, which stores assertions about a particular domain, and the $\mathcal{TBox}$, which is the actual ontology. The $\mathcal{TBox}$ contains both the terms used by an ontology for a specific domain and the axioms which govern how $\mathcal{ABox}$ assertions relate to one another [5]. gives examples of $\mathcal{TBox}$ definitions, and gives examples of $\mathcal{ABox}$ assertions about individuals such as MARY and PETER. It is appropriate to equate individuals in an $\mathcal{ABox}$ to records in a database when performing data related processing using Description Logics. The $\mathcal{ABox}$ in OWL has the open world assumption (OWA) [61] that if a statement is *unknown* it has not been falsified. In contrast, the closed world assumption (CWA) states that if a statement is not known to be true, it is false. These assumptions are related to defaults, which resolve ambiguities and missing values in a closed world system, but cannot be assumed in the open world. New developments in

inductive methods have been proposed to close the gap between CWA defaults and any ambiguities they introduce in the open world [5].

**Table 3. $\mathcal{TBox}$ Example**

| Concept | | Definition |
|---|---|---|
| Father | $\equiv$ | Male ⊓ ∃hasChild.Person |
| Mother | $\equiv$ | Female ⊓ ∃hasChild.Person |
| MotherWithoutDaughter | $\equiv$ | Mother ⊓ ∀hasChild.¬Female |
| PersonWithoutChildren | $\equiv$ | Person ⊓ ∀hasChild.⊥ |
| FatherWithAtMostTwoFemaleChildren | $\equiv$ | Father ⊓ ≤ 2 hasChild.Female |
| FatherWithAtMostTwoChildren | $\equiv$ | Father ⊓ ≤ 2 hasChild |

**Table 4. $\mathcal{ABox}$ Example (Roles on Individuals)**

| Description | Definition |
|---|---|
| MARY is a Mother without daughters | MotherWithoutDaughter(MARY) |
| MARY has a child PETER | hasChild(MARY, PETER) |
| MARY has a child PAUL | hasChild(MARY, PAUL) |
| PETER is a Man with a child | Father(PETER) |
| PETER has a child HARRY | hasChild(PETER, HARRY) |
| PETER is an only child | ≤ 1 hasChild(PETER) |
| ENGINE is part of a CAR | CAR ⊑ ∃hasPart.ENGINE |
| The owner of CAR owns (its part) ENGINE | ∃owns.CAR ⊑ ∃owns.ENGINE |

To ensure greater tractability of OWL, OWL 2 has been proposed, which is based on the *SROIQ* extension, and allows strict partial order on roles in the form of role-hierarchies [62] as per . OWL 2 is comprised of three sub-languages called profiles[21], each with a set of properties suitable for different tasks.

The *EL* profile[22] allows for polynomial time algorithms, and is suitable for large ontologies, where expressivity is sacrificed for performance of reasoning tasks. This increase in performance is achieved by limiting *EL* ontologies to include tractable class restrictions and axioms. These include selecting elements from a finite set, intersections of such sets, subsumption relations, equivalent and disjoint class axioms, transitivity and reflexivity. Excluded intractable class restrictions include defining

[21] http://www.w3.org/TR/owl2-overview/#Profiles
[22] http://www.w3.org/TR/owl2-new-features/#OWL_2_EL

unbounded sets, cardinality restrictions, unions, disjoint properties, negation and a universal quantifier.

The *QL* profile[23] has been created to allow conjunctive queries to be processed in LogSpace time, and is meant to be used with existing relational database systems, where $\mathcal{ABox}$ assertions are represented by records in a database, and the $\mathcal{TBox}$ is a set of relational query like structures.

Finally, the *RL* profile[24] is a syntactic subset of OWL 2 that allows for polynomial time reasoning algorithms, to be used on rule-extended database systems. This partial axiomatization of OWL 2 semantics places restrictions on the *RL* profile. These restrictions aim to avoid inference of individuals that do not already exist in the $\mathcal{ABox}$, and to avoid non-deterministc reasoning. The *RL* profile is suitable to express the OWL 2 rules derived by the algorithm in this thesis. Specifically, these rules represent the information contained within the $\mathcal{ABox}$ of the local and target systems. See Appendix 2 on page 91 for specific examples of OWL 2 syntax utilized by this thesis.

## 2.4 Existing Approaches

Systems such as the Quick Ontology Matcher (QOM) [18] build similarity measures between ontology concepts based on similarity of terms. Direct matches such as $Person_L$ from the local (*L*) ontology and $Person_T$ from the target (T) ontology are identified as exact matches. Indirect matches through synonyms, such as $Car_L$ and $Automobile_T$ are matched as well. Through a top-down iterative process, the matches made in previous iterations are used to build additional term *anchors*[25]. By using anchors such as these, additional measures are utilized, specifically features of ontological entities expressed in RDFS[26], since ontologies matched by QOM must be represented in RDFS. The characteristics include the uniform resource locators (URL) to identify unique entity labels. Structural characteristics are also utilized to find further matches, including inherited and direct properties, sub-properties, parent-properties as well as concepts, sub-concepts, and super-concepts of previously found matches. The OWL *owl:sameAs* relationship is utilized to identify explicit matches between already matched terms. Logical inconsistencies are used to remove any conflicting lexical matches.

[23] http://www.w3.org/TR/owl2-new-features/#OWL_2_QL
[24] http://www.w3.org/TR/owl2-new-features/#OWL_2_RL
[25] Refer to the Glossary on page 112 for the definition of *anchor*.
[26] RDFS: (Resource Description Framework Schema) A general-purpose language, in XML syntax, for representing assertions (or more generally information) on the Web, as proposed by W3C.

After scoring each match, QOM implements a filtering step based on a set threshold to disregard insignificant matches.

Systems such as QOM which use a variety of techniques, essentially create a set of criteria and methods called Parameterizable Alignment Methods (PAM) [22] which can be used to match ontologies. Systems such as APFEL [22], or Alignment Process Feature Estimation and Learning, can utilize the work done by systems like QOM and derive additional matches. APFEL allows users to iteratively validate matches manually. Once they are validated, it uses various machine learning algorithms such as decision trees, neural networks, and support vector machines to infer best PAM parameters to use, and iteratively applies these parameters to derive further matches.

Xu et. al. [23] also uses a collection of matching techniques, but to analyze record data in addition to lexical information provided by schema labels. The system first (1) finds terminology relationships, such as synonyms and hypernyms with the use of WordNet [20]. It then 2) identifies data-value characteristics such as string length and alphanumeric ratios. 3) Domain specific characteristics are then utilized such as expected strings and predefined formats (e.g. phone number), through the use of regular expressions. Similarly to QOM, after using concept *anchors* based on weights assigned in steps 1 - 3, the system then 4) looks for structural similarities, such as inherited concepts. Matching is done by creating a decision tree to rank each matching technique [24]. Based on the decision tree rules, a particular technique is applied to identify the best presumed match.

Ichise et. al. [63] applies support vector machines (SVM) to select best matching techniques between concepts. The techniques, called "match properties" are similar to the ones already listed in this chapter, including lexical and structural matching techniques. 1) "Word" similarity includes lexical similarities such as prefixes, suffixes, edit distances and n-grams. 2) "Word-list" similarity applies to phrases that can be written in different forms, but have the same intended meaning, such as "Social Sci" and "Social Science." 3) "Concept hierarchy" identifies hierarchical similarities, such as parent-to-child relations between concepts, based on lexical anchors derived in 1 and 2. It then 4) uses "structure similarity" to take into account the entire ontology, not just direct parent and child nodes as in 3), and implements "graph based similarity" measures, with an emphasis on parent nodes. The algorithm calculates scores for each concept-property pair, and the SVM selects closest property matches. Concepts are deemed similar if they have similar property measures for a particular technique.

Ontologies have also been used to build decision trees, where an ontology is used to classify concepts. As demonstrated in the development of the Ontology-driven Decision Tree (ODT) algorithm [64], ontologies provide the hierarchical relation is-a (*Coke* is-a *Soda*) to link records in the data with super-classes in the ontology. For example, a record in a dataset identified as *Coke* can be linked to another record identified as *Milk*, through ontological relationships[27] *Coke* $\subseteq$ *Soda* $\subseteq$*Beverage* and *Milk* $\subseteq$ *DairyBeverage* $\subseteq$ *Beverage*. ODT considers an attribute's information gain, and modifies the decision tree by inserting the super-class of each record from the ontology as a sub-node, instead of the actual records, as with the traditional decision tree algorithm. A similar approach to ODT was used in combination with user ratings to develop a recommender system called SemTree [65]. The advantage in using an ontology is that the key factor of the building process, the information-gain used to associate an attribute to a concept, is based on the attribute's association with that concept in relation to other concepts in the ontology, and not just the attribute's data value.

Kieler et. al. match specific geospatial locations [66] and specific rivers [67] based on records from different systems, captured by different sensors, and most importantly at different scales. The algorithm builds decision trees from the provided records, which share a similar database schema with the target database. Based on the generated rules, sufficient similarities are found in the data, which identify a particular location or object. The rules are able to correctly classify that location or object, despite being built from information that is not exact in nature. Because the database schemas are sufficiently similar, Kieler et. al. is able to utilize all attributes, by allowing the decision tree algorithm's property selection process to chose the attributes to be included. Because the ontologies targeted by this thesis may have irregular similarities, as described in the following chapter, multiple 2-dimensional decision trees (utilizing 2 attributes at a time) are built, and compared as a group.

The current work on including inductively derived information has focused on classification of assertions ($\mathcal{ABox}$) in a Description Logic (DL) knowledge base, by associating uncertainty to the ontology's terminology ($\mathcal{TBox}$). BayesOWL [31] has been proposed to perform automatic ontology mapping [36] by associating probabilities with text based information, and using Jeffrey's Rule to propagate those probabilities. Text documents are classified using a classifier such as Rainbow[28], and probabilities are assigned using the conditional probability table (CPT) process. BayesOWL creates probabilities for OWL Description Logics by converting a DL to a directed acyclic graph (DAG), and assigning probabilities to each edge using a CPT, for two types of nodes; concept nodes and L-nodes

[27] The subset $\subseteq$ symbol is used here instead of subsumption to reflect the syntax in the original paper.
[28] http://www-2.cs.cmu.edu/~mccallum/bow/rainbow

(logical relations) [31]. As an example, the CPT probability for an *equivalent* L-node between concepts $c_1$ and $c_2$, is 1.0 if [ $(c_1 \wedge c_2) \vee (\neg c_1 \wedge \neg c_2)$ ], and 0.0 otherwise, while the probability of a *complement* L-node is 1.0 if [ $(\neg c_1 \wedge c_2) \vee (\neg c_1 \wedge \neg c_2)$ ], and 0.0 otherwise. Tools such as OWL-CM [37] have begun looking at how similarity measures and uncertainties in the mapping process can be improved to increase access correspondences between text ontology entities.

In this thesis, we utilize a similar process to ODT to associate records with ontological concepts. We expand on this approach to associate records with the entire ontological parent chain. We use this concept-to-record association as classification criteria for decision trees. The expanded parent chain is used to identify classification[29] criteria at each hierarchical level of the ontology, in a bottom-up fashion. Different classification criteria are identified at each level, and utilized by a machine learning algorithm, mainly decision trees, to produce criteria for concept *anchors*. This approach differs from other methods in that it produces *anchors* based primarily on database records, instead of lexical information extracted from labels used by the ontology author. Similarly to the work cited here, we use these local *anchors* to find additional structural characteristics between other local *anchors* and matched them to target *anchors*.

[29] As noted in the Glossary under *class*, classification refers to the meaning used in machine learning literature.

# Chapter 3. Methodology

## 3.1 Approach

System integration deals with merging two separate systems, or sub-systems, to work together in achieving a single goal, or produce a single output [1]. Often this involves merging business logic through an Application Programming Interface (API), or by a much more intricate process where individual application libraries are merged or interlinked, to produce a hybrid system. This may involve integrating not just the supporting infrastructure, but the databases processed by the individual systems. Database integration deals with integrating database structure tables, columns, and possibly rows (instance matching) [2][3]. Semantic integration deals with integrating two systems where not just the format of information is considered, but also the intended meaning of the elements to be matched [68][69].

As with many database integration techniques [3][70][71], this thesis uses machine learning measures to derive properties about the data. Unlike database integration however, the matching will not be done between a local and a target database schema, but between a local and target ontology, and is called ontology matching [10].

**Definition 1: (Local System Terms).** The ontology matching algorithm described in this thesis is dependent on a locally available database $\mathcal{DB}_L$ and a *local* ontology $O_L$ that represents semantic relationships between local concepts $\{LC\}$. Data records in $\mathcal{DB}_L$ are made up of attributes $\{da\}$ in the set $DA_L$. Formally the local system is made up of:

| | |
|---|---|
| $\mathcal{DB}_L$ | ::= $\mathcal{DB}_L$ is the local database used to classify concepts in $O_L$ by building decision trees. |
| $f_L$ | ::= $f_L$ is the number of attributes in a normalized version of $\mathcal{DB}_L$ |
| $DA_L$ | ::= $DA_L$ is a set of data attributes used in $\mathcal{DB}_L$, defined as $\{da_0, \ldots, da_{fL}\}$. |
| $O_L$ | ::= $O_L$ is a local ontology being matched to an external target ontology. |
| $LC$ | ::= $LC$ is a local concept in $O_L$. |
| $\text{Level}_L(LC)$ | ::= $\text{Level}_L(LC)$ is the minimal number of edges $LC$ is away from the root (*owl:Thing* in OWL ontology) class, or root when $O_L$ is a taxonomy. |

The Semantic Integration process strives to match $O_L$ to an external *target* ontology $O_T$, by matching their individual concepts.

**Definition 2: (Target System Terms).** The target system being matched has a different database $\mathcal{DB}_{\mathrm{T}}$, which contains similar information to $\mathcal{DB}_{\mathrm{L}}$. Similar in the sense that both databases contain records capturing information of the same or similar dataset. For example, $\mathcal{DB}_{\mathrm{L}}$ may contain weather phenomena recorded by weather stations in Africa, and $\mathcal{DB}_{\mathrm{T}}$ of weather phenomena recorded in South America. Formally the target system is made up of:

| | |
|---|---|
| $\mathcal{DB}_{\mathrm{T}}$ | ::= $\mathcal{DB}_{\mathrm{T}}$ is the target database used to classify concepts in $O_T$ by building decision trees. |
| $f_T$ | ::= $f_T$ is the number of attributes in a normalized version of $\mathcal{DB}_{\mathrm{T}}$. |
| $DA_T$ | ::= $DA_T$ is a set of data attributes used in $\mathcal{DB}_T$, defined as $\{da_0, \ldots, da_{fT}\}$. |
| $O_T$ | ::= $O_T$ is a target ontology being matched to a local ontology $O_L$. |
| $TC$ | ::= $TC$ is a concept in $O_T$. |
| $\mathrm{Level}_{\mathrm{T}}(TC)$ | ::= $\mathrm{Level}_{\mathrm{T}}(TC)$ is the minimal number of edges $TC$ is away from the root (*owl:Thing* in OWL ontology) class, or root when $O_T$ is a taxonomy. |

## 3.2 Semantics and Machine Learning

As described in previous chapters, there is a great deal of interest in ontology matching, and utilizing lexical and structural information extracted from an ontology. There is also a growing interest in incorporating machine learning, uncertainty and probability to assist with automatic or semi-automatic ontology matching. To further this effort, this thesis proposes matching ontologies not based on lexical information of their terms, but on the information they represent. For this reason this thesis considers ontologies that are associated with a dataset that can be mined with machine learning algorithms, specifically the classification algorithm for decision trees. The decision tree algorithm needs a column that holds a set of class values, which are used to classify records. This is usually associated with the record type, a particular condition the record represents or an associated conclusion represented by the record; usually this class value is already part of the data, or embedded in the application.

Ontologies, however, can be concepts at a higher level of abstraction, described by a single record, collection of records, or a combination of column values. For example, different types of precipitation can be described by the combination of water fall recorded in milliliters, the current temperature in Celsius, visibility in kilometers, or wind velocity in kilometers per hour: low water fall, temperature

above 0 degrees Celsius, high visibility and low winds may represent *Rain*. Lower temperatures and less visibility may represent *Snow*. These four pieces of information can be stored in different columns of a single record, in multiple records, or multiple tables, within a single database.

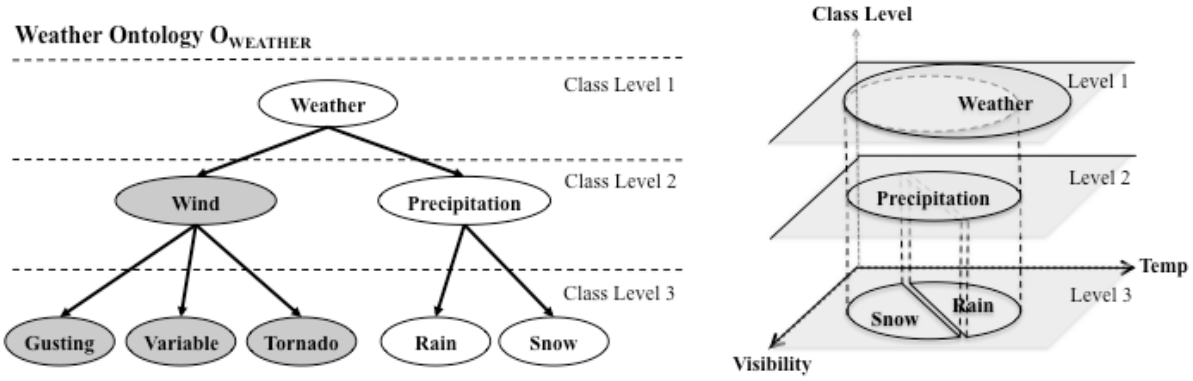



**Figure 2. A sample (a) *Weather* Ontology $O_W$ with (b) 2-dimensional data clusters at each class level for the attributes *Temperature* (*Temp*) and *Visibility*.**

Consider the partial Weather Ontology $O_{WEATHER}$ in Figure 2 (a). Here we have a top level concept *Weather* at level 1 which subsumes concepts *Wind* and *Precipitation*. At level 2 *Wind* subsumes *Gusting*, *Variable* and *Tornado*, while *Precipitation* subsumes *Rain* and *Snow*. Once each of these concepts has been assigned records from $\mathcal{DB}$, we can plot these records[30] onto a 2-dimensional plane. The discussion on dimensionality and number of attributes is deferred to section 3.5. In Figure 2 (b) we plot data-points for the attributes *Visibility* and *Temperature* (*Temp*) for records assigned to *Weather* on Level 1, *Precipitation* on level 2, and finally *Rain* and *Snow* on level 3.

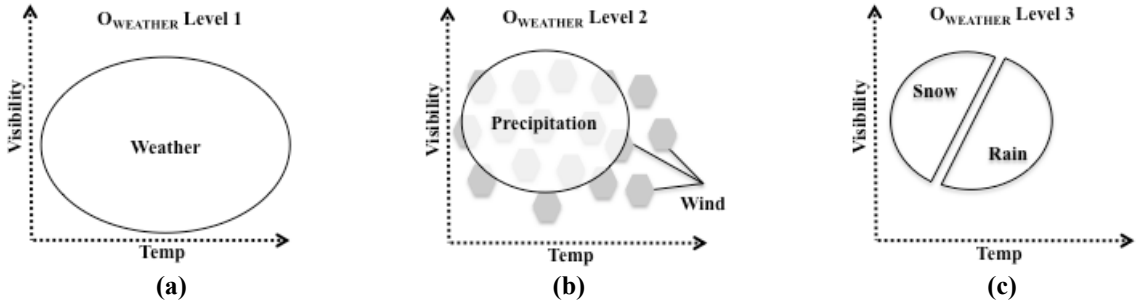



**Figure 3. A sample (a) Weather Ontology $O_{WEATHER}$ with (b) 2-dimensional data clusters at each class level for the attributes Temperature (Temp) and Visibility (Vis).**

[30] See section 3.3 for examples of data records and associated concepts.

We now look at each class level individually. Figure 3 shows the same information as Figure 2 (b) with each concept and class level in Figure 3 (a), (b) and (c). Figure 3 (b) also shows data points for the Wind concept as small grey clusters, to illustrate an important fact about which attributes to use when building decision trees. The *Visibility* and *Temperature* attributes clearly represent the Weather concept in Figure 3 (a) as a distinct data cluster, as well as give a clear distinction[31] between Rain and Snow in Figure 3 (c). There is no such distinction between Precipitation and Wind in Figure 3 (b) however, and these may not be the best attributes with which to distinguish *Precipitation* and *Wind* records using decision trees.

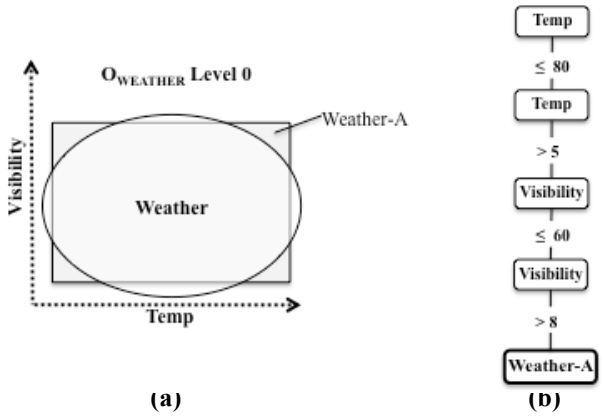



**Figure 4. A 2-dimensional representation of the (a) area and (b) decision tree rules classifying the *Weather* concept using attributes *Temperature* (*Temp*) and *Visibility*.**

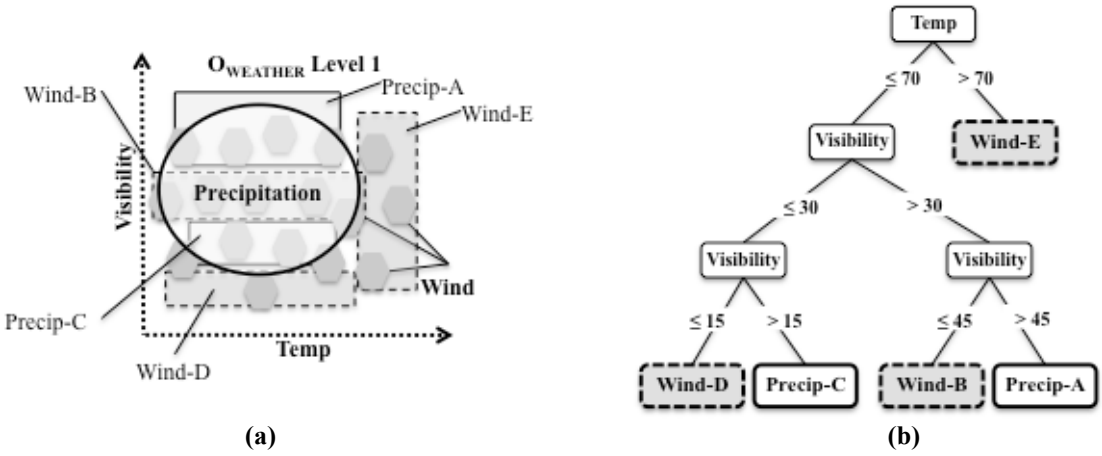



**Figure 5. A 2-dimensional representation of the (a) area and (b) decision tree rules classifying the *Wind* and *Precipitation* concepts using attributes *Temperature* (*Temp*) and *Visibility*.**

[31] These examples are for demonstration purposes only, and are not meant to be representative of an actual clear distinction between Rain and Snow.

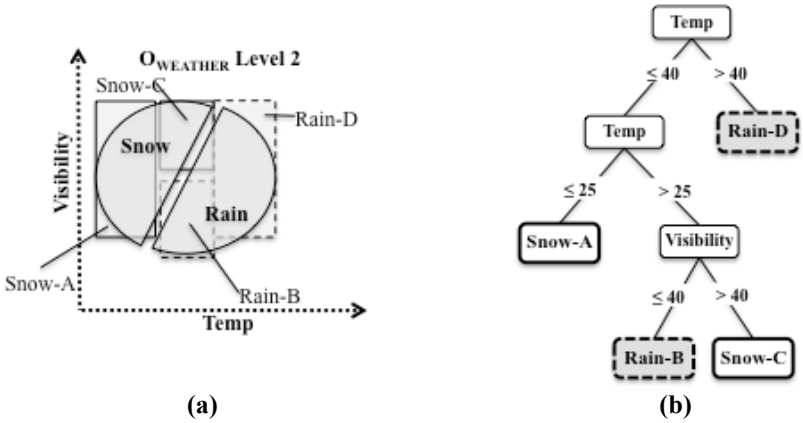



**Figure 6. A 2-dimensional representation of the (a) area and (b) decision tree rules classifying the Rains and *Snow* concepts using attributes *Temperature* (*Temp*) and *Visibility*.**

To demonstrate this point further, we show what a decision tree for each concept may look like. In Figure 4 (a) we again see data-points for the *Weather* concept with a rectangle representing the area classified by the decision tree rule as *Weather-A* using the attributes V*isibility* and *Temperature*, and the decision tree itself in Figure 4 (b)[32]. Note that this area is a leaf node of the decision tree in Figure 4 (b) that we will call a *granule*[33], and this granule represents a range of data-points in the form of a rule. In Figure 5 (a) we see granules attempting to identify the areas representing *Wind* (dashed sides) and *Precipitation* (solid sides). Because there is no clear distinction between these two concepts using the *Visibility* and *Temperature* attributes, the decision tree created in Figure 5 (b) is not a good model for *Wind* or *Precipitation*. This tree is represented by granules in Figure 5 (a), where it is clearly visible that some of the granules with dashed sides (*Wind*) are clearly overlapping with the main oval area representing *Precipitation*. These granules classify encompassed data-points as *Wind*, but have a low probability of correctness since many of the points are in fact *Precipitation*. The same is true for the granules with solid sides representing *Precipitation* classification rules that encompass *Wind* data-points.

We again demonstrate the areas encompassing data-points associated with *Rain* and *Snow* in Figure 6 (a) using the attributes *Visibility* and *Temperature*. This time we include granules that represent classification rules for *Rain* and *Snow* with the corresponding decision tree in Figure 6 (b). Here there is a clear distinction between these two concepts. As a result the decision tree's classification rules and resulting granules have a high probability of correctness.

[32] Note that this is an oversimplified example for demonstration purposes only.

[33] A granule represents a data cluster, identified by a leaf node in a decision tree. It contains components usually identified with a decision tree leaf node, as per Definition 3 on page 36 and Definition 4 on page 37.

## 3.3 Database Preparation

We now describe the procedure for associating database records with ontological concepts. The resulting dataset will allows us to perform supervised learning to build a decision tree model of the records that an ontology is meant to describe semantically. In order to achieve this, the database $\mathcal{DB}$ must be associated with the ontology $O$ in such a way that the ontological concepts are used as classification criteria for the machine-learning algorithm. This preparation is similar to the ODT [64] system in that it associates an ontological concept to a column classifying a particular database record. We extend this notion by representing an entire hierarchical branch and each concept in that branch with that record in the database.

The records that $O$ describes are represented by a tuple, which we assume is stored in a database $\mathcal{DB}$. For a relational database, multiple tables must be denormalized. In this process, all attributes and relationships are brought into a single table, with logical and hierarchical relations being represented as database attributes in a single row. It is important to represent concepts at equivalent levels[34] by the same column, with different classes as separate columns. The reason is that the decision tree algorithm will use a particular column as the classification criterion. For example, in order to classify all direct sub-classes of *owl:Thing*, the "Class Level 0" column will be used for all records in the database. Therefore it is important that all sibling concepts are identified together at the same level, by the same column. Representing ontology levels is demonstrated in Figure 7 and Figure 11 in section 3.5 below. For example, consider the following two branches in Table 5. They are represented as database records 4 and 5 in Table 6, where *owl:Thing* is the ontology root concept.

**Table 5. Concept hierarchy levels as database columns**

| *Record #* | *Class Level 0* | *Class Level 1* | *Class Level 2* | *Class Level 3* | *Class Level 4* |
|---|---|---|---|---|---|
| ***4*** | *owl:Thing* | ⊒ *WeatherEvent* | ⊒ *VisibilityEvent* | ⊒ *ScatteredCloudLayer* | |
| ***5*** | *owl:Thing* | ⊒ *WeatherEvent* | ⊒ *Measured* | ⊒ *TemperatureEvent* | ⊒ *CurrentDewPoint* |

Table 6 below demonstrates this type of hierarchy as a denormalized table with all database attributes. Notice that the "Class Level 0" column contains *owl:Thing* in all rows. This represents the root node of the ontology's hierarchical representation. Multiple super-classes are represented by a duplication

[34] It is not required for levels to align when matching concept clusters and rules across ontologies, only when initially creating the decision trees, since the parent-to-child concept classification is done in isolation from the rest of the tree.

of records 8 and 9 representing the subsumption relationships (*Showers* $\sqsubseteq$ *WeatherDescriptor) and (Showers* $\sqsubseteq$ *PrecipitationEvent*). Also, a record may represent multiple concepts. This is achieved by again duplicating records with different class values. Records 4 and 5 are equal in property values, and represent (*ScatterCloudLayer* $\sqsubseteq$ *VisibilityEvent*) and (*CurrentDewPoint* $\sqsubseteq$ *TemperatureEvent*), and records 6 and 7 represent (*HighVariableWindEvent* $\sqsubseteq$ *VariableWindEvent*) and (*Cumulonimbus* $\sqsubseteq$ *CloudType*).

**Table 6. Normalized Data Sample**

| Instance # | Visibility Direction English | Temperature1 | Temperature2 | Hourly Precipitation | Visibility Statute Miles | Humidity Percent | Wind Direction English | Weather Intensity | Wind Speed MPH | Wind Gust MPH | Has Station ID | Inches Of Mercury | Class Level 0 | Class Level 1 | Class Level 2 | Class Level 3 | Class Level 4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ? | 28 | 24 | ? | 3 | ? | NW | Light | 2 | 5 | KSYN | 4.6 | owl:Thing | Weather Event | Visibility Event | Scattered Cloud Layer | ? |
| 2 | ? | 4 | 1 | 10 | 4 | 35 | N | Medium | 5 | 6 | YPKG | 3.6 | owl:Thing | Weather Event | Visibility Event | Scatter Cloud Layer | ? |
| 3 | N | 51 | 14 | 29 | 1 | 32 | NE | ? | 1 | 2 | KEVB | ? | owl:Thing | Weather Event | Visibility Event | Scattered Cloud Layer | ? |
| 4 | NE | 7 | 1 | 3 | 8 | 25 | SE | Light | 0 | 0 | KSYN | 4.2 | owl:Thing | Weather Event | Visibility Event | Scattered Cloud Layer | ? |
| 5 | NE | 7 | 1 | 3 | 8 | 25 | SE | Light | 0 | 0 | KSYN | 4.2 | owl:Thing | Weather Event | Measured | Temperature Event | Current Dew Point |
| 6 | NE | 5 | 5 | 28 | 4 | 83 | WE | Medium | 2 | 4 | PADL | ? | owl:Thing | Weather Event | Wind Event | Variable Wind Event | HighVariable WindEvent |
| 7 | NE | 5 | 5 | 28 | 4 | 83 | WE | Medium | 2 | 4 | PADL | ? | owl:Thing | WeatherQualifier | Cloud Type | Cumulonimbus | ? |
| 8 | SW | 14 | 6 | 64 | 13 | 45 | N | Heavy | ? | 0 | KAJG | ? | owl:Thing | Weather Event | Current Weather Event | Precipitation Event | Showers |
| 9 | E | 17 | 10 | 17 | 20 | 54 | W | Heavy | 4 | 5 | EYKA | 1.0 | owl:Thing | Weather Descriptor | Showers | ? | ? |
| 10 | S | 89 | 58 | 99 | 4 | 47 | NE | Light | 4 | 7 | KRMG | 3.6 | owl:Thing | Weather Event | Measured | PressureEvent | Altimeter Setting |
| 11 | W | 75 | 55 | 43 | 8 | 38 | S | Heavy | 3 | 8 | KSAC | 3.2 | owl:Thing | Weather Descriptor | Thunderstorm | ? | ? |
| 12 | NE | 8 | 1 | 3 | 7.2 | 86 | SE | Medium | 1 | 3 | EYKA | 2.7 | owl:Thing | WeatherQualifier | Cloud Type | Cumulonimbus | ? |
| 13 | SW | 8 | 1 | 600 | 7.2 | 14 | E | Medium | 4 | 8 | KSAT | 8.0 | owl:Thing | WeatherQualifier | Cloud Type | Cumulonimbus | ? |
| 14 | NW | 7 | 1 | 30 | 8.4 | 41 | W | Light | 2 | 5 | KBWG | ? | owl:Thing | WeatherQualifier | Cloud Type | Standing Lenticulor Or Rotor Clouds | ? |
| 15 | S | 7 | 1 | 31 | 8.4 | 40 | NW | Heavy | 1 | 3 | KALW | ? | owl:Thing | Weather Descriptor | Blowing | ? | ? |
| 16 | NE | 7 | 1 | 44 | 8.4 | 20 | NE | Heavy | 2 | 2 | KSAT | ? | owl:Thing | Weather Descriptor | Blowing | ? | ? |
| 17 | E | 10 | 46 | 32 | 4 | ? | ? | ? | 3 | 1 | KALW | 3.1 | owl:Thing | Weather Event | Measured | PressureEvent | Altimeter Setting |
| 18 | E | 10 | 46 | 32 | 4 | ? | ? | ? | 3 | 1 | KALW | 3.1 | owl:Thing | Weather Descriptor | Blowing | ? | ? |
| 19 | ? | ? | 3 | 62 | 7 | ? | ? | ? | ? | 0 | KBUR | 7.2 | owl:Thing | WeatherQualifier | Cloud Type | Altocumulus Castellanus | ? |
| 20 | NE | ? | 3 | 18 | 13 | ? | ? | ? | ? | 0 | KBUR | ? | owl:Thing | WeatherQualifier | Cloud Type | Altocumulus Castellanus | ? |

## 3.4 Attribute Matching

Before utilizing database records for semantic integration through classification, attributes of the local and target databases must first be matched. Once this is complete, the algorithm can use data-mining to find similarities and match concepts. While there are many methods for matching database attributes [70], as an example we describe a simple statistical matching process here, and give its representation in OWL 2 syntax in Example 9 as part of Appendix 2.

To match numeric data attributes we use the Java implementation of a statistical library[35] provided by WEKA to calculate an attribute's *standard deviation*, *mean*, *maximum*, and *minimum*. We then create rules that define a range for each of these *values* and for each attribute. We do not use the original *values* for numeric ranges, but instead expand the range's boundaries, using an arbitrarily chosen factor of *0.25*. The value of this factor is dependent on the precision required, and the fluctuation of the attributes being matched. We choose an arbitrary value of *0.25* for our example to demonstrate the algorithm, and recognize that a more in-depth analysis would be required for individual implementations of the attribute matching process. The expanded minimum for a range is (*value* × *0.75*), and the expanded maximum is (*value* × *1.25*). For example, if the *standard deviation* of a local property $da_0$ is $\sigma_0 = 10.0$, it will match the standard deviation $\sigma_1$ of a target property $da_1$ in the range $7.5 \leq \sigma_1 \leq 12.25$.

If a data attribute is nominal, we use the terms extracted from WordNet [20] 3.0 to create a set of *terms* which represent that attribute's nominal range. We then consider local attribute $da_a$ and target attribute $da_b$ equivalent if any of the *key terms* in $da_a$ and $da_b$ match. *Key terms* are identified as the actual values found in the database for an attribute, plus any extracted terms shared by all original values. For example, suppose that the database attribute *Direction* contains amongst its records the value "N" which stands for *north*. "N" is associated in WordNet with the intended term "compass point" as well as the unintended term "nitrogen", amongst others. The term "compass point" is consistently extracted from WordNet for other *Direction* terms such as "S" for *south*, "W" for *west*, "E" for *east*, and so on. This indicates that "compass point" is a good term to match a field representing *Direction*, and is considered as a *key term*, while "nitrogen" is not. Here, the *key terms* for *Direction* will be "N", "S", "E", "W", "compass-point". Note that this is not meant to be an exhaustive definition, and that more terms are actually associated with directions in WordNet 3.0.

[35] weka.experiment.Stats

## 3.5 Concept Matching

Once local and target attributes are matched, the concept matching process can begin. This process works by finding decision tree rules that overlap one another. With every additional property utilized by the decision tree, the dimensionality of the search space between the rules being matched increases as well. This creates very strict criteria for finding a match. Such criteria can be beneficial when the systems being matched use the same database, and very similar schemas [66][67], but not systems in which similar data can be interpreted and represented differently in the database as well as the ontology. To make the matching algorithm more general and increase the probability of finding a match between different systems, the dimensionality of the search space must be reduced. For this reason, this algorithm creates a set of 2-dimensional search spaces for each data cluster, by building multiple decision trees based on two attributes at a time. The generality of 2-dimensional decision trees also prevents over-fitting in any single tree to a specific combinations of multiple attributes. Situations where this reduction in dimensions in the search space may not work, occurs when there is a great deal of relations in different combinations of attributes. For example, if it is known that attributes $da_0$, $da_1$ and $da_2$ always appear together in order to identify some concept $A$, it would be necessary to include 3-dimensional trees in the search space. If all combinations of 3 attributes are necessary, computing power would need to be sufficiently increased to handle this.

As a collection of possible rules, each decision tree and combination of attributes increases the probability of finding a match between two heterogeneous systems that show some similarities. The ideal solution would be to include every combination of attributes, producing $(n \times (n - 1) / 2)$ combinations for $n$ attributes. Another solution would be to use an attribute evaluation algorithm, perhaps based on entropy, to choose the attributes that contribute most to the classification process, and use those attributes to create several 2-dimensional trees. The latter method is very sensitive to differences in the distribution of data in $\mathcal{DB}_{\mathrm{L}}$ and $\mathcal{DB}_{\mathrm{T}}$. For the real-life test in Chapter 4, we included every combination of all attributes, and disregarded any decision trees that had a single and no branches. For the evaluation test using the weather ontology that up to 1 million records per test, we chose yet another attribute selection method, due to the computation limitation of the test machines used. We chose pairs of consecutive attributes to cover as many combinations as possible, without exponentially growing the number of trees being compared. For example the four attributes $da_0$, $da_1$, $da_2$, $da_3$, will produce four 2-dimensional trees for $da_0$ vs. $da_1$, $da_1$ vs. $da_2$, $da_2$ vs. $da_3$ and $da_3$ vs. $da_0$. See section 4.7 in the Results and Analysis chapter for a further discussion on the issues affecting the performance of the algorithm presented here.

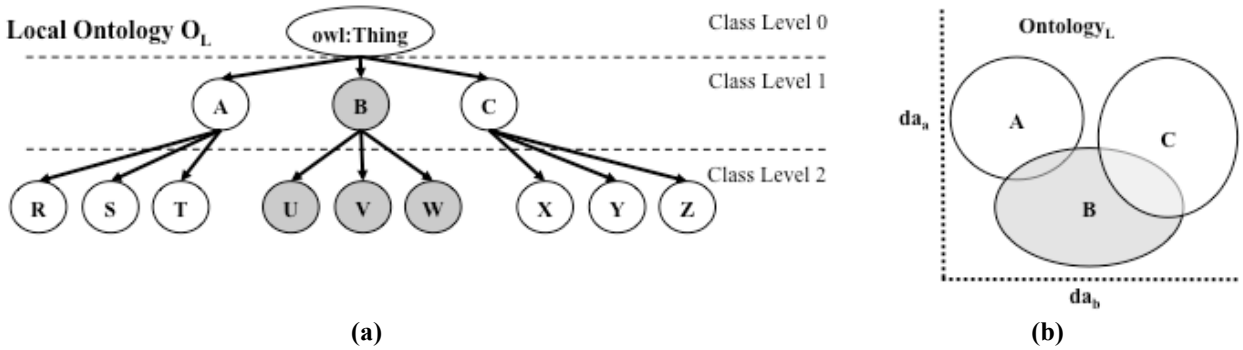



**Figure 7. A sample taxonomic Local Ontology ($O_L$) with class levels, and a 2-dimensional cluster for data attributes $da_a$ and $da_b$.**

A demonstration of a taxonomy is shown in Figure 7, where part (a) shows a hierarchical OWL ontology which we'll call our local ontology $O_L$, with sub-class relations between concepts, and part (b) shows the clusters which represent Level 1 of the ontology and data distribution for data attributes $da_a$ and $da_b$, in the associated database $\mathcal{DB}_L$. The top concept is *owl:Thing*[36], which is the root of the ontology and the super-class of all other concepts. At level 1 the ontology contains three concepts: $LC_A$, $LC_B$, $LC_C$ which are sub-classes of *owl:Thing*. At level 2 the ontology has nine more concepts: $LC_R$, $LC_S$, $LC_T$ which are sub-classes of $LC_A$; $LC_U$, $LC_V$, $LC_W$ which are sub-classes of $LC_B$; and $LC_X$, $LC_Y$, $LC_Z$ which are sub-classes of $LC_C$. To integrate the ontology with its associated data in $\mathcal{DB}_L$, each leaf concept of the ontology is associated with all the records that represent that concept, where the ontology concept being represented is he last "Class" column with a value, as per Table 6. Once the records are assigned, we can visually display the associated data values on a 2-dimensional plane with the two attributes for the axes, as illustrated in Figure 7 (b). See sections 3.2 and 3.3 for a detailed description of combining data and an ontology into a 2-dimensional plane.

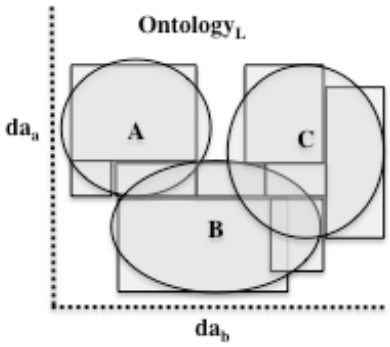



**Figure 8. $O_L$ Level 1 granulation.**

[36] *owl:Thing* is the root node of every OWL ontology, and every class is implicitly subsumed by *owl:Thing* (http://www.w3.org/TR/2004/REC-owl-guide-20040210/#DefiningSimpleClasses)

With the use of decision trees, these clusters can be broken down further into smaller granules defined by rules derived by the decision trees. Figure 8 demonstrates such granules at level 1, where the decision trees classify concepts $LC_A$, $LC_B$, and $LC_C$.

The clusters in Figure 7 can be further split on level 2, visualizing clusters for concepts $LC_R$, $LC_S$, $LC_T$, $LC_U$, $LC_V$, $LC_W$, $LC_X$, $LC_Y$, and $LC_Z$, as indicated in Figure 9 below.

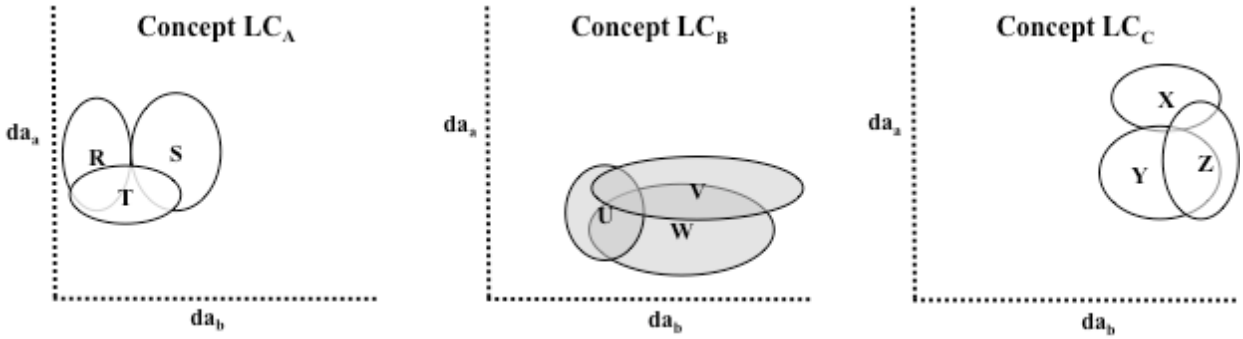



**Figure 9. 2-dimensional, Level 2 sub-class clusters for $O_L$.**

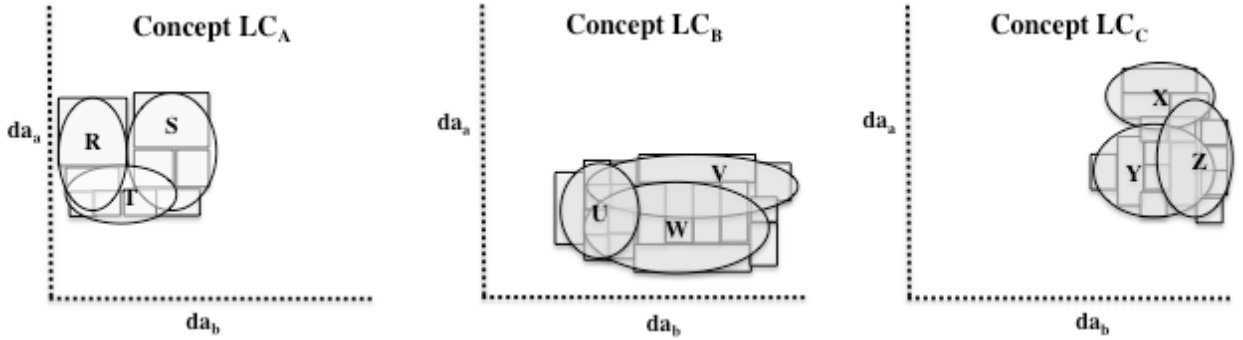



**Figure 10. $O_L$ Level 2 granulation.**

Similarly to Figure 8, level 2 concepts can be classified with decision trees, which create much smaller and more specific granules, as demonstrated in Figure 10.

## 3.6 Semantic Granule Building

With granules, associated decision trees and rules build we are ready to look at how this can be used to match a local ontology to a target ontology. In Figure 11 below, we have an example of a target ontology $O_T$, which has all the same ground terms but is structured slightly differently from $O_L$: $TC_R$, $TC_S$ and $TC_T$ are sub-classes of $TC_A$; $TC_U$ is a sub-class of $TC_N$; $TC_V$ and $TC_W$ are sub-classes of $TC_P$;

$TC_X$, $TC_Y$ and $TC_Z$ are sub-classes of $TC_C$. Assume the associated database $\mathcal{DB}_T$ has some similarities with $\mathcal{DB}_L$. Assume also that the data clusters for all ground terms are similar, and overlap when plotted and clustered on a 2-dimensional plane.

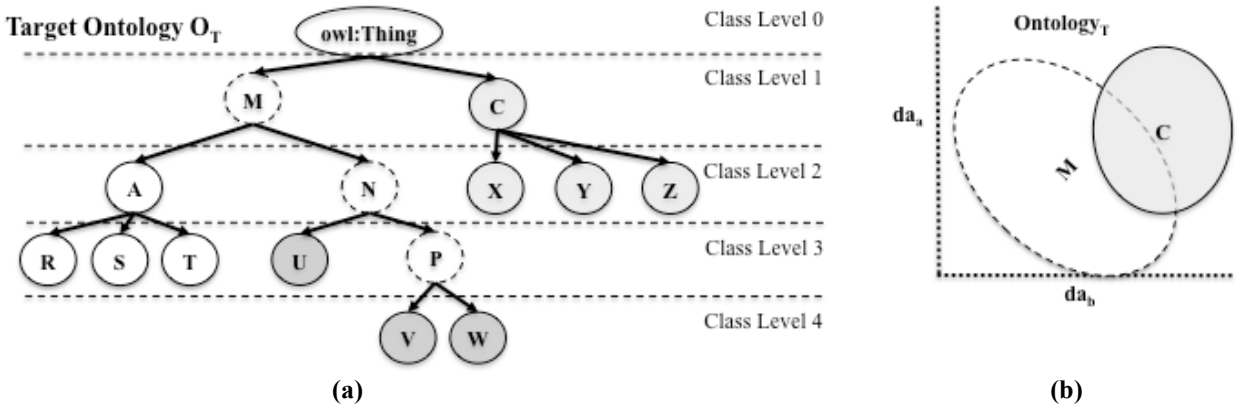



**Figure 11. Standard Target Ontology ($O_T$) with class levels, and a 2-dimensional clusters.**

The key differences, identified by dotted lines, between $O_L$ and $O_T$ are that the structure of the sub-class relationships is different. The sub-classes of $A$ are still $R$, $S$ and $T$, but while $LC_A \sqsubseteq_{\text{direct}}$ *owl:Thing*, $TC_M$ is introduced in $O_T$, where $TC_A \sqsubseteq_{\text{direct}} TC_M \sqsubseteq_{\text{direct}}$ *owl:Thing*. $TC_C$ is structured in the same way as $LC_C$. $LC_B$ is split into two different sub-classes; mainly $TC_N$ and $TC_P$, where $TC_P \sqsubseteq_{\text{direct}} TC_N \sqsubseteq_{\text{direct}} TC_M$. Matching $LC_C$ to $TC_C$ is the simplest task by looking only at the sub-classes. $LC_A$ can be matched to $TC_A$ in a similar way. $LC_B$ however has no exact match in $O_T$ because it is split between $TC_N$ and $TC_P$. We can now classify concepts at lower levels of $O_T$, and derive granules for its ground terms, mainly $TC_R$, $TC_S$, $TC_T$, $TC_U$, $TC_V$, $TC_W$, $TC_X$, $TC_Y$, and $TC_Z$.

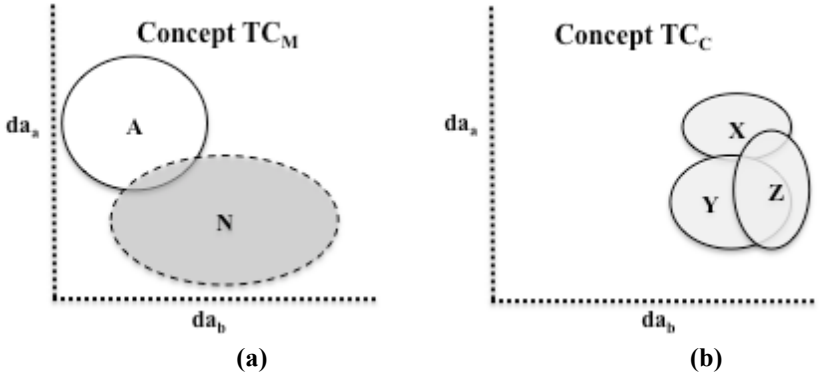



**Figure 12. 2-dimensional, Level 2 sub-class clusters for $O_T$.**

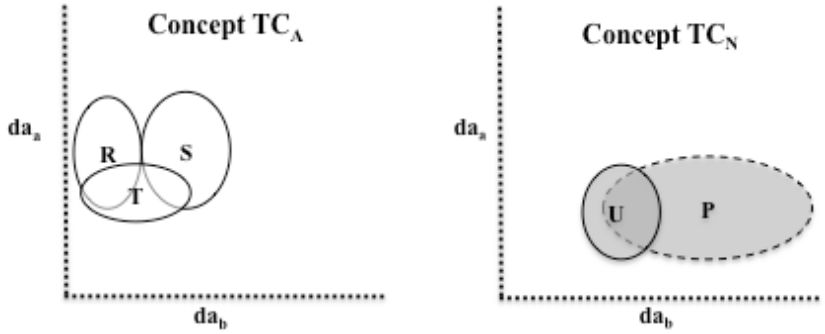



**Figure 13. 2-dimensional, Level 3 sub-class clusters for $O_T$.**

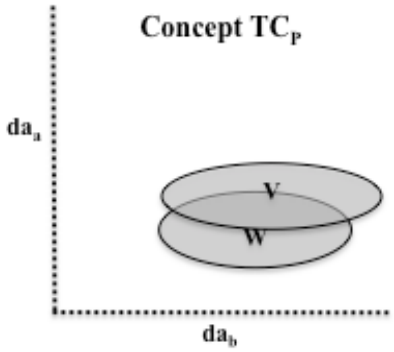



**Figure 14. 2-dimensional, Level 4 sub-class clusters for $O_T$.**

Figure 11 to Figure 14 have projected onto a 2-dimensional plane, clusters that represent data classified under ontological concepts in $O_T$. This is different from usual classification criteria, which is either a column in the database or a condition embedded in an application. Both the database schema and application logic are implementation details, whose main purpose is to process information efficiently. The ontology on the other hand defines semantic relationships between not just the records in $\mathcal{DB}_T$, but also the abstract concepts these records are meant to represent in the real world. These abstract concepts may be independent from the implementation details and functional constraints placed on a database and application logic. While it may be correct to assume that the tightly integrated data concepts in the local system do not necessarily share similarities with an external target system, it may be possible that both the local and target ontologies are representing real life entities, such as *Person*, *Home*, *Dwelling*, *Rain*, *Snow*, and so on. The data driven approach in this thesis is meant to identify such similarities and utilize them for the ontology matching process.

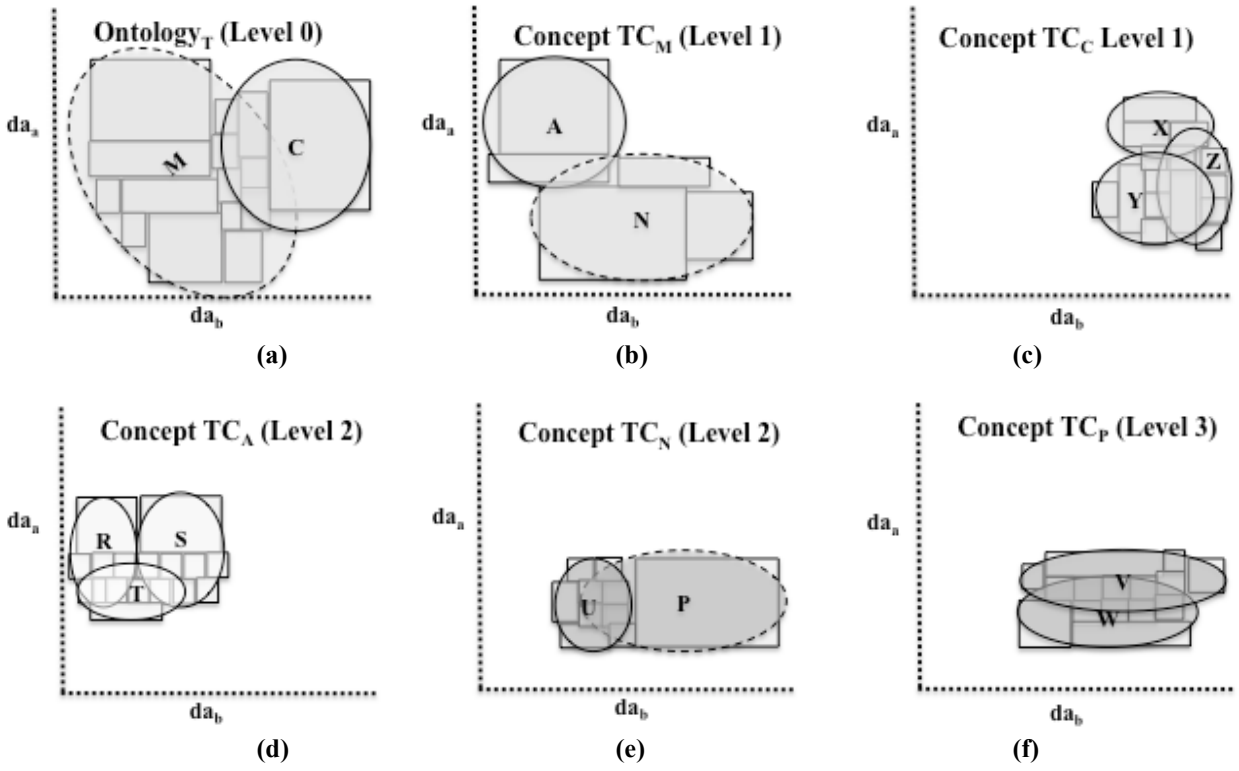



**Figure 15. Complete granulation (Levels 0 – 3) of $O_T$.**

To ensure the intended meaning of concepts in a system are matched, even though they may differ in their structure but not ontological representation, we again create clusters of each 2-dimensional space. As with the local system, $O_T$ is associated with records in $\mathcal{DB}_T$ , decision trees are built to classify concepts in $O_T$, and the resulting rules that represent data-point ranges are represented as granules. Each of the granules in Figure 8 and Figure 10 for $O_L$ and Figure 15 for $O_T$, represents a single rule generated by the algorithm. Figure 16 illustrates the decision tree classifying records based on $O_L$ concepts $LC_R$, $LC_S$ and $LC_T$, which are sub-classes of $LC_A$, and utilize data attributes $da_a$ and $da_b$.

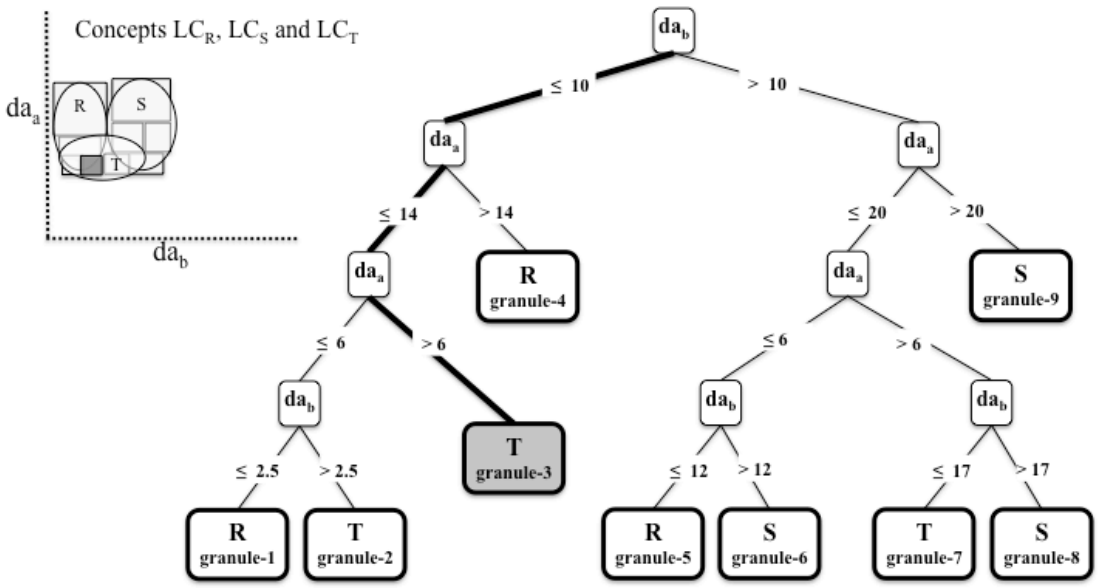



**Figure 16. Decision tree classification with 2 numeric data attributes for sub-classes of $LC_A$.**

In Figure 16, each leaf node classifies a particular sub-class of $LC_A$. As Figure 9 (a) and Figure 13 (a) demonstrate, clusters in the plotted graphs overlap each other, which results in the classifications being mixed, and a clear distinction between $LC_R$, $LC_S$, or $LC_T$ is not present. Instead, the decision tree algorithm creates a set of rules, which identify granule-3 (Figure 16 - inset axis), where each granule is a branch in the decision tree. This branch is a set of conditions that identify a particular granule for each class being classified by a leaf node. All granules for sub-classes of $LC_A$ and $TC_A$ are illustrated in Figure 10 and Figure 15 (d). In Figure 16, the highlighted area (inset axis) and branch identify a granule for concept $LC_T$. The rules for this $LC_T$ granule, defined as:

$$[Lg_T\,(\text{granule-3})] \equiv [\,(\,da_b \leq 10\,) \wedge (\,da_a \leq 14\,) \wedge (da_b > 6\,)\,]$$

are a conjunction of the conditions represented by the branch leading from the root of the tree to [$Lg_T$ (granule-3)].

**Definition 3: (Ontology Concept Granule).** Each ontological concept is identified with records stored in its respective database. Based on the record, each concept is broken down into granules. Formally these are:

$Lg_i$ ::= $Lg_i$ is a local granule representing a leaf node in a decision tree built by classifying concepts in $\mathcal{DB}_{\mathrm{L}}$, where $i$ is the index or label uniquely identifying $Lg_i$ within $O_L$.

$Tg_j$ ::= $Tg_j$ is a target granule representing a leaf node in a decision tree built by classifying concepts in $\mathcal{DB}_{\mathrm{T}}$, where $j$ is the index or label uniquely identifying $Tg_j$ within $O_T$.

Let "A" be either "L" or "T" for local or target input respectively. Each concept is made up of a set of granules:

$$AC_z \equiv (Ag_0 \vee \ldots \vee Ag_k \vee \ldots \vee Ag_n)$$

where each granule $Ag_k$ implies a unique $AC_z$ concept by:

$$Ag_k \Rightarrow_{\mathrm{G}} AC_z$$

Because each granule stems from a leaf node in the decision tree, it also possesses several statistical properties, as described in **Definition 1**, most notably the distribution of correctly classified records by each leaf node. The derived granule $Ag_k$ is comprised of several components, as defined in **Definition 4** below.

**Definition 4: (Granule Components).** Let "A" be either "L" or "T" for local or target input respectively. Each granule *Ag* implies *AC,* identified by $Ag \Rightarrow_G AC$.

*Ag.attribs* ::= Data attributes utilized in the decision trees used to classify *Ag* as *AC*, where $Ag.attribs = \{da_a , da_b\} \in DA_A$ and $DA_A$ is the set of all data attributes in $\mathcal{DB}_A$.

*Ag.rule* ::= Conjunction of conditions which define the granule's area within the 2-dimensional property space of a numeric data attributes $da_a$ and a nominal data attribute $da_b$:

$$[\ (da_a \diamondsuit l_0) \wedge \ldots \wedge (da_b \diamondsuit l_n)\ ] = Ag.rule$$

where $da_b$ has only a single value, and

$$[\ (da_a \diamondsuit l_0) \wedge (da_a \diamondsuit l_1) \wedge (da_b \in \{ l_3, l_4 , l_5, l_6\})\ ] = Ag.rule$$

where $da_b$ has multiple values defining a nominal range, $\diamondsuit \in \{\geq , \leq , =\}$, and $l_0 , \ldots , l_n$ are literals used in the condition for $da_a$ and $da_b$. In case of nominal attributes, a nominal range is generalized with the use of WordNet [20]. This list is discussed in section 3.8.

*Ag.data-points* ::= $da_a$ and $da_b$ data-points identifying outer edges of the area covered by *Ag.rule*:

$$[\ (da_a = l_0) \wedge \ldots \wedge (da_b = l_n)\ ] = Ag_k.data\text{-}points$$

*Ag.cover* ::= Total records classified inside *Ag* by the decision tree.

*Ag.good* ::= Total records *correctly* classified inside *Ag* by the decision tree.

*Ag.Pr* ::= The probability that the decision tree leaf node represented by $Ag_k$ was correct in classifying the associated $AC_z$, where:

$$Ag.Pr = \frac{Ag.good}{Ag.cover} \quad (1)$$

## 3.7 Granule Rule Matching

Up to this point, it has been illustrated how to classify records based on ontological concepts, and build rules identifying granules using the decision tree algorithm. Using the derived granules, with associated rules and data-points, the matching process is now illustrated, by matching the local concepts $LC_R$, $LC_S$ and $LC_T$, to target concepts $TC_R$, $TC_S$ and $TC_T$. Figure 17 illustrates the overlapping granules for $LC_A$ and $TC_A$ sub-classes.

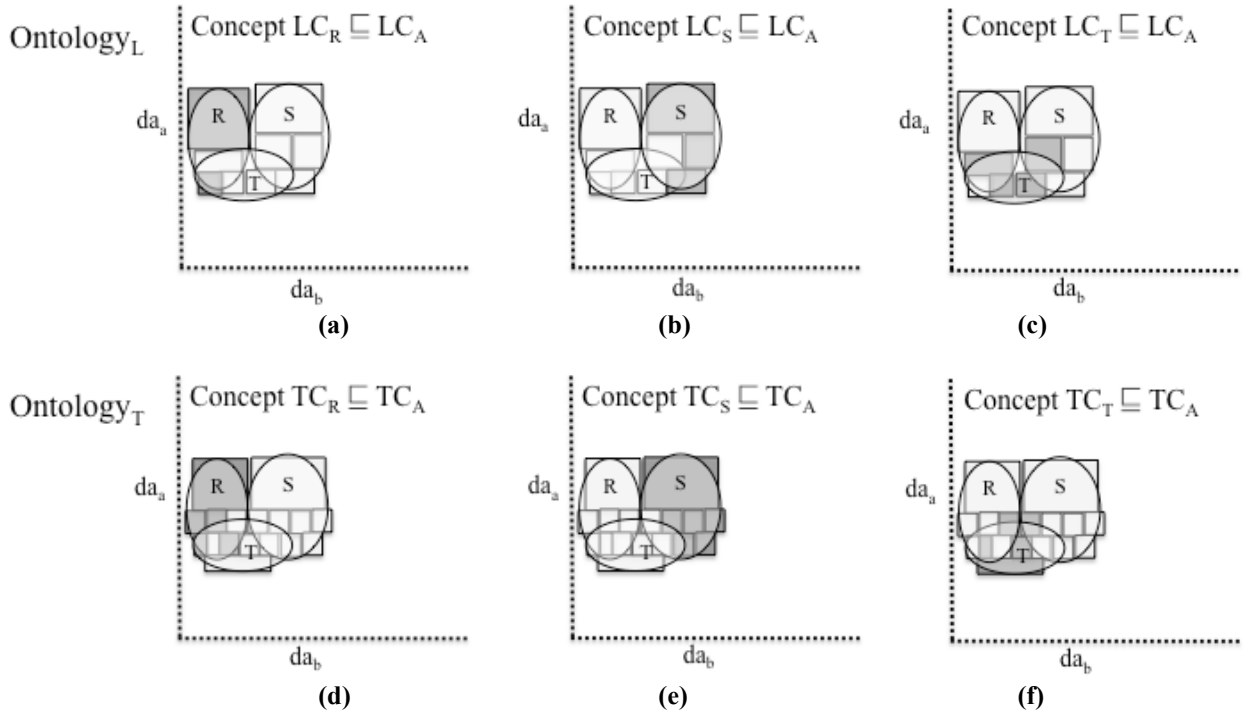



**Figure 17. $LC_A$ and $TC_A$ granule matches between $R$ (a) ≡ (d), $S$ (b) ≡ (e), and $T$ (c) ≡ (f).**

As indicated by the shaded areas in Figure 17, there is a visible overlap between the areas covered by these granules. The matching algorithm described by this thesis works by identifying these overlapping regions, and rating them. Figure 18 and Figure 19 illustrate how the matching process works, and how the reasoner views the granules in terms of data-points and rules.

The reasoner utilized in this thesis is HermiT [40], a freely available OWL 2 reasoner. To identify overlapping granules, the reasoner needs two components: 1) rules identifying a range, and 2) data-points which fall within that range. The rules are a semi-direct translation of decision tree branches into OWL 2 axioms. The data-points are outer most points defined by the range in the rule. Infinite conditions are left out, leaving an open boundary. For example, if the condition on $da_a$ is only $da_a > l_n$, the rule will match any data-point greater then or equal to $l_n$. Rule translation is semi-direct because the rules are automatically made more inclusive for matching purposes, in two ways. Firstly, while numeric ranges created by decision trees are based on $\{\leq , >\}$, the OWL 2 translations built by the algorithm are based on $\{\leq , \geq\}$, which are more inclusive. Note that this is not a limitation of OWL 2 as it is fully capable of handling any of facets using $\{\leq , \geq , < , >\}$. Secondly, the boundary values of the ranges are changed, expanding the covering area defined by the rules, and decreasing the area between data-points by a user defined factor called *RULE_CLUSTER_BUFFER*. The generalized

boundaries are given in Equations (2) to (5). Figure 18 below illustrates the inclusivity measure for two granules, $Lg_0$ (granule-0) and $Tg_1$ (granule-1) with numeric values.

$$\text{rule.upper}_{gen} = (\text{upper} + ((\text{upper} - \text{lower}) \times \text{factor}) \quad (2)$$

$$\text{rule.lower}_{gen} = (\text{lower} - ((\text{upper} - \text{lower}) \times \text{factor}) \quad (3)$$

$$\text{data-points.upper}_{gen} = (\text{upper} - ((\text{upper} - \text{lower}) \times \text{factor}) \quad (4)$$

$$\text{data-points.lower}_{gen} = (\text{upper} - ((\text{upper} - \text{lower}) \times \text{factor}) \quad (5)$$

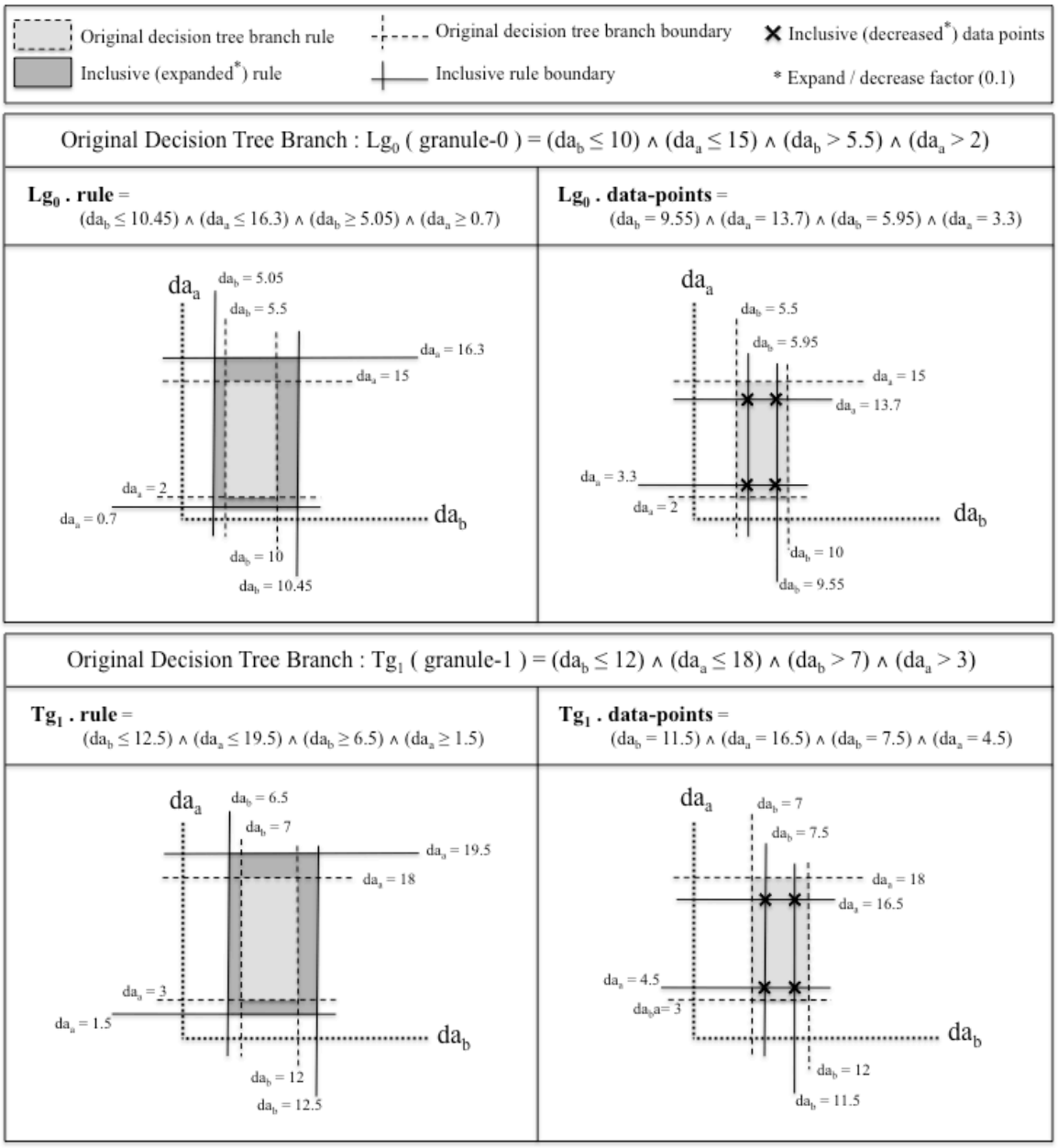



**Figure 18. Decision Tree branch conversion to rules and data-points using numeric attributes $da_a$ and $da_b$.**

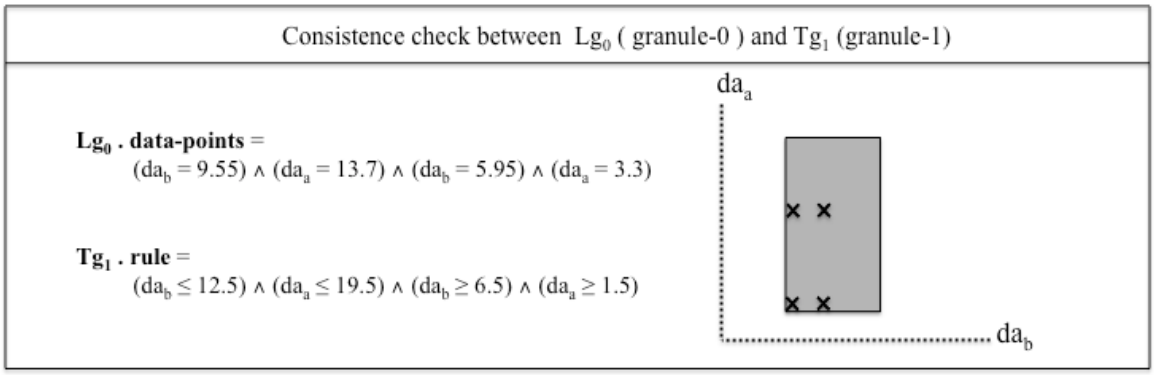



**Figure 19. Granule matching example using numeric attributes $da_a$ and $da_b$.**

## 3.8 Nominal Property Rules

Up to now, we have dealt with rules and data-points for numeric attributes only. Next, we demonstrate how nominal attributes are converted from decision tree branches to rules and data-points, and describe the matching process.

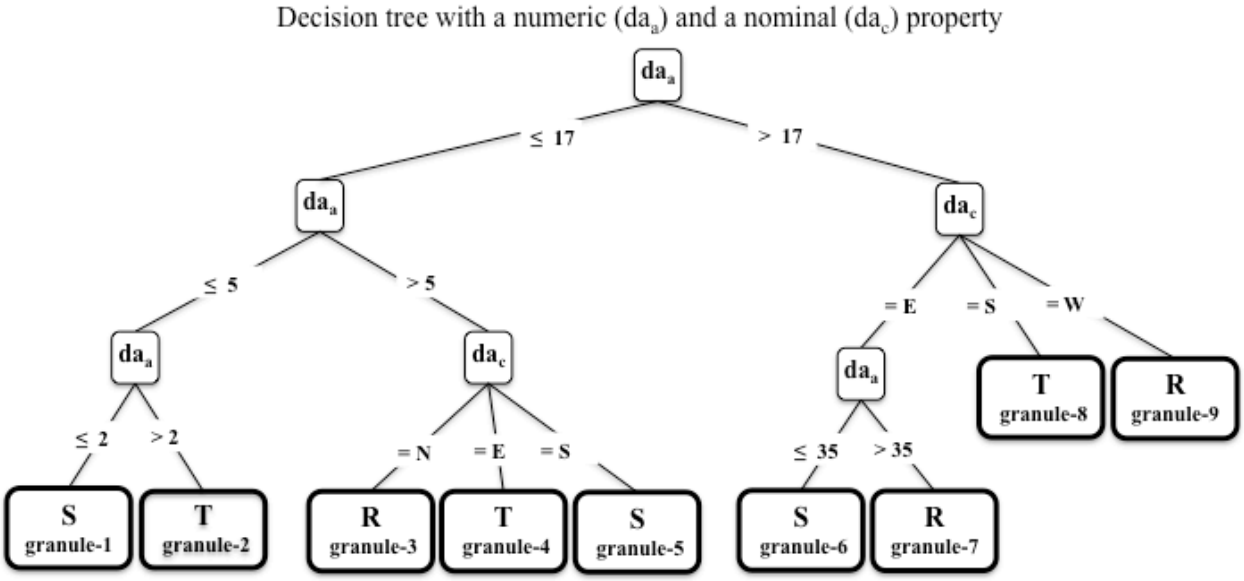



**Figure 20. Decision tree classification, utilizing a numeric ($da_a$) and a nominal ($da_c$) property.**

In Figure 20, a decision tree classifies concepts $LC_R$, $LC_S$ and $LC_T$ that are sub-classes of $LC_A$, using a numeric and a nominal property, $da_a$ and $da_c$ respectively. With numeric attributes, data values included in rules and data-points have been derived directly from the records in a database. While numerical values depend on their numerical ranges to create clusters, which they can be matched to, nominal values present a different challenge. A single letter, term, phrase, or a set of terms, cannot be used to represent the intended meaning of that term. For example, one ontology might use "N," and

another “North”, or “Northbound”. For this reason, WordNet [20] was incorporated to build rules which not only contain the actual values for a database property in records identified as some concept, but also some related terms extracted from WordNet. This generalizes the rules to find matches with other nominal “data-points”. The rules and “data-points” for nominal data attributes are a disjunction of possible values, made up of the original value and extracted WordNet terms. For example, an original value might be $da_{b\text{-}original}$ = (“N”), and the one included in granule rules and data-points might be $da_{b\text{-}generalized}$ = (“n”, “compass north”, “northward”, “360 degrees”}

The terms extracted from WordNet represent a generalized meaning of that data attribute value. It is important to avoid including any specific terms which may be associated with the terms used only in a particular local system, as this will limit the generality of matches. The following WordNet sets are utilized:

- Adjective Satellite, and all associated synonyms.
- Adjectives, and all associated synonyms.
- Nouns, and all associated synonyms, hypernyms, instance hypernyms, member holonyms, substance holonyms, associated topics, topic members, regions and region members.
- Verbs, and all associated synonyms, hypernyms, troponyms, verb groups, entailments, associated topics and regions.

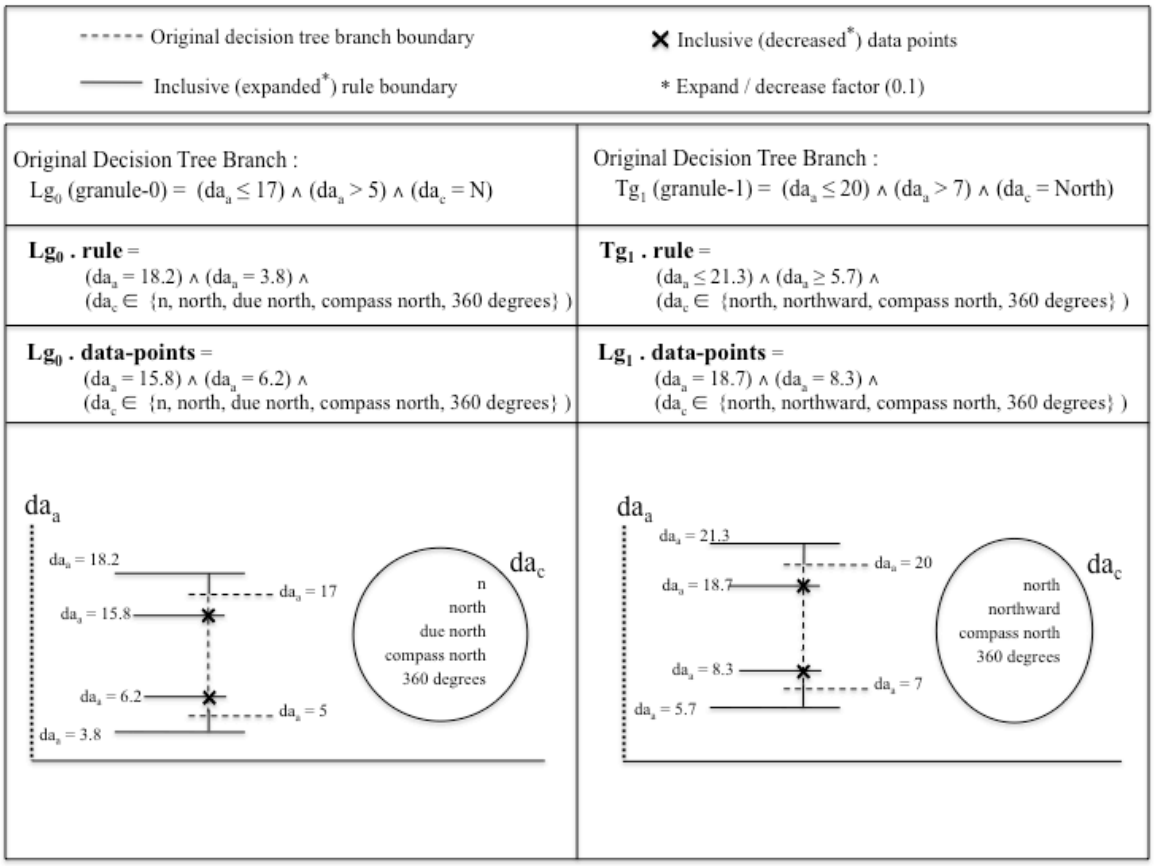



**Figure 21. Decision Tree branch converted to rules and data-points with nominal and numeric attributes $da_a$ and $da_c$.**

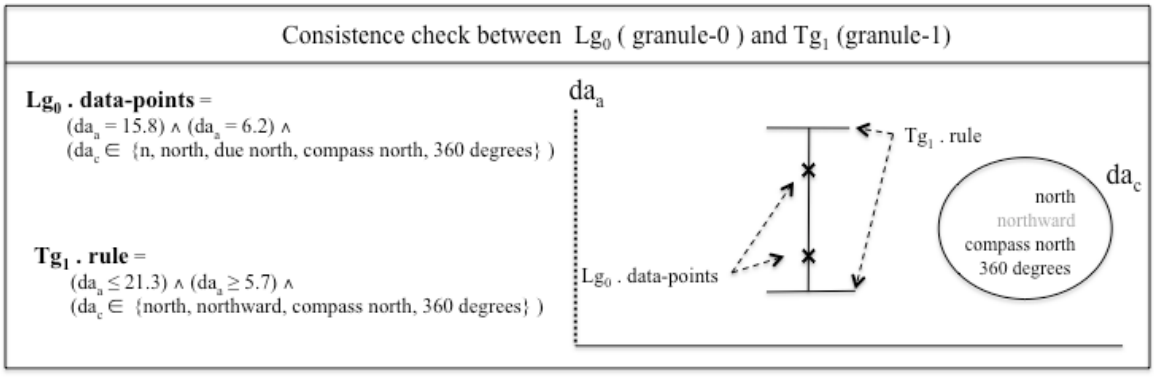



**Figure 22. Granule matching example using the numeric attribute $da_a$ and nominal attribute $da_c$.**

Figure 21 lists the conversion of nominal decision tree branches into granule rules and data-points. Figure 22 illustrates how such rules and data-points can be matched by the algorithm.

## 3.9 Matching Algorithm

Once the granules $\{Lg_k\}$, with associated $Lg_k.rule$ and $Lg_k.data\text{-}points$ are built, these are made available to external parties using OWL 2 syntax. This section defines the matching algorithm, and matching schemes used to rank and derive further structural matches, based on granules and the local and target ontology.

**Definition 5: (Points To Rule Match P2RM).** Given a local granule *Lg*, and a target granule *Tg*, consistency between two granules depends on an overlap between *Lg.data-points* and *Tg.rule,* as illustrated in Figure 19 and Figure 22. Specifically, *if the attributes used by a local granule Lg.attribs and a target granule Tg.attribs are all the same, and if the data-points of a local granule Lg.data-points are covered by the rules of a target granule Tg.rule, then there exists a relation between the ordered pair (Lg, Tg), identified as P2RM(Lg, Tg)*. All *P2RM* relations between the granules of *LC* and *TC*, where $Lg \Rightarrow_G LC$ and $Tg \Rightarrow_G TC$ , are members of the set P2RM$(LC, TC)_{\text{set}}$.

**Definition 6: (Concept Match).** A "Concept Match" is a match between a local and a target concept. The match criterion is a successful overlap between a local and a target granule, as per **Definition 5**, where the local (target) granule implies the local (target) concept being matched. Specifically, *ConceptMatch is a ternary relation between a local concept LC, a target concept TC and the set* $P2RM(LC, TC)_{set}$ *if there exists at least one P2RM(Lg , Tg) relation in* $P2RM(LC, TC)_{set}$ *such that* $Lg \Rightarrow_G LC$ *and* $Tg \Rightarrow_G TC$. This relation is identified as *ConceptMatch*(*LC, TC,* P2RM$(LC, TC)_{\text{set}}$ ), or *ConceptMatch*(*LC*, *TC*, $\text{P2RM}_{\text{set}}$*)* for short.

**Definition 7: (Local Concept To Target Concept Match C2CM).** $\text{C2CM}_{\text{set}}$ is a set of all *ConceptMatch* relations as per **Definition 6**, ultimately creating a set of all matches between local and target concepts.

Once a set of local granules is generated for a local ontology, this ontology can be potentially matched to an external target ontology, based on the associated target granules. The granule matching algorithm in **Definition 8**, attempts to find any matches between a local (*LC*) and target (*TC*) concept. Matching concepts requires first matching any overlapping local granule's *Lg.data-points* and a target granule's *Tg.rule*, as illustrated in Figure 19 and Figure 22, and defined in **Definition 5**. An overlap occurs if the data points defined in $Lg_i.data\text{-}points$ are within the ranges defined in $Tg_j.rule$, as defined by **Definition 5**, and is called a "Points to Rule Match," identified by the *P2RM* relation between a local and target granule. A successful *P2RM* relation is then associated with the local and target

concepts that each granule implies, as per **Definition 6**. The resulting ternary relation *ConceptMatch* becomes a member of $C2CM_{set}$, as per **Definition 7**, which represents a set of all matched concepts.

We now describe the Granule Matching Algorithm in **Definition 8** below. The algorithm first (1) associates all previously established matches between local and target data attributes, $DA_L$ and $DA_T$ respectively by adding the EquivalentDataProperty[37] axiom for each local and target data property pair. On step (2 – 16) it iterates through each local granule $\{Lg_i\}$ over $i$, and the associated $Lg_i.data\text{-}points$. For each $Lg_i.data\text{-}points$, on steps (4 - 14) the algorithm iterates through all target granules $\{Tg_j\}$ over $j$, and the associated target rules $Tg_i.rule$, and ensures only one $Lg_i.data\text{-}points$ axiom and $Tg_j.rule$ axiom is present in $O_w$. In step (7) the consistency between $Lg_i$ and $Tg_j$ is checked as per **Definition 5** by inserting a *EquivalentClasses*[38] axiom stating that $Lg_i$ and $Tg_j$ are equivalent, and in in step (8) verifying whether this axiom did not cause $O_w$ to be unsatisfiable. If $O_w$ is still satisfiable, the two granules $Lg_i$ and $Tg_j$ are matched in step (9) by creating the "Points To Rule Match" relation $P2RM(Tg_j, Lg_i)$, as per **Definition 5**. In step (11) the *EquivalentClasses* axiom previously insert in step (7) is then removed, along with the target granule rule $Tg_j.rule$ axiom in step (12) previously inserted in step (6).

[37] The *EquivalentDataProperty* axiom is an OWL 2 axiom which states that two data properties a EquivalentDataProperty are equivalent, as per the definition in Table 27 on page 91.

[38] The *EquivalentClasses* axiom is an OWL 2 axiom which states that two concepts are equivalent, as per the definition in Table 25 on page 91.

**Definition 8: (Granule Matching Algorithm).** During the $Lg_i.data\text{-}points$ and $Tg_j.rule$ matching process, the "work" ontology $O_W$ is used to verify the two granules are consistent with each other for all local and target granules, where:

$O_W$ ::= $O_w$ is a "work" ontology used for reasoning with axioms in the matching process.

and the algorithm is:

```
1)  Add all EquivalentDataProperty[39] Axioms into O_w
2)  Foreach Local Points Axiom as Lg_i.data-points
3)      Add Lg_i.data-points into O_w
4)      Foreach Target Rule Axiom as Tg_j.rule
5)         If Lg_i.attribs are equivalent to Tg_j.attribs
6)           Add Tg_j.rule into O_w
7)           Add EquivalentClasses[40](Lg_i.data-points, Tg_j.rule) Axiom into O_w
8)           If O_w is satisfiable (no unsatisfiable classes exist)
9)               Create the P2RM(Lg_i, Tg_j) relation, as per Definition 5.
10)          End
11)          Remove EquivalentClasses(Lg_i.data-points, Tg_j.rule) from O_w
12)          Remove Tg_j.rule from O_w
13)        End
14)     End
15)     Remove Lg_i.data-points from O_w
16) End
17) Foreach Target Concept as TC
18)     Foreach Local Concept as LC
19)        If P2RM_set ≠ ∅, as per Definition 6
20)          Create the ConceptMatch(LC, TC, P2RM_set) relation, as per Definition 6.
21)          ConceptMatch(LC, TC, P2RM_set) is a member of C2CM_set, as per Definition 7.
22)        End
23)     End
24) End
```

[39] The *EquivalentDataProperty* axiom is an OWL 2 axiom which states that two data properties a EquivalentDataProperty are equivalent, as per the definition in Table 27 on page 91.

[40] The *EquivalentClasses* axiom is an OWL 2 axiom which states that two concepts are equivalent, as per the definition in Table 25 on page 91.

After the individual local granules ($Lg_i$) have been matched with all consistent target granules ($Tg_j$), in steps (17 - 24) the target concepts are iterated, and for each target concept *TC*, local targets are iterated, in order to create the ternary *ConceptMatch* relation between local and target concepts *LC* and *TC*, and the set of associated *P2RM* relations $P2RM_{set}$, as per **Definition 6**. In step (21) the resulting *ConceptMatch* relation becomes a member of the set $C2CM_{set}$ which contains all matches between local and target concepts, as per **Definition 7**.

## 3.10 Concept Match Ranking Measures and Motivation

Up to this point, the algorithm to perform concept matching has been described. We now present the motivation for the methods by which each match is ranked. After a high level introduction, sections 3.11 and 3.12 present formal definitions of concept-to-concept match rankings. As a running example, we define the local and target ontologies in Figure 23, which demonstrate how the *Precipitation* concept is defined by each ontology.

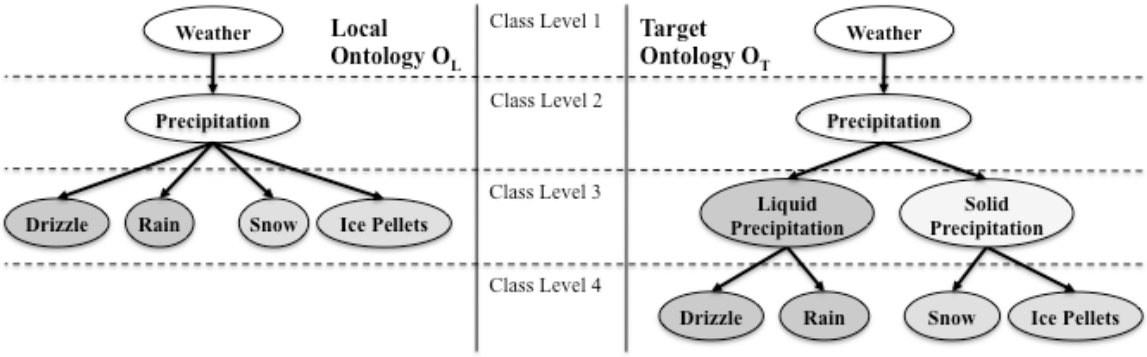



**Figure 23. Local and Target Ontologies defining the Precipitation concept.**

Once granules are matched, as per **Definition 5**, we must rank each individual granule-match to infer the best concept-matches. Recall from **Definition 4** on page 37, that each granule represents a leaf node in a decision tree, and that each granule has a probability associated with that leaf node. This probability is the number of *correctly* classified records in the leaf node divided by the number of *all* records classified by that leaf node. For example, a leaf node may have classified 20 records as $Rain_T$, but only 15 of them are actually $Rain_T$ in the database. The probability of this node being correct would be *15 / 20 = 0.75*. This value, which we call *Lg.Pr* and *Tg.Pr* for local and target granules respectively and define in **Definition 4** on page 21, gives us a ranking of individual granules.

Because there may not be a clear distinction between *Rain* and *Drizzle*, there may be an overlap between some *Rain* and *Drizzle* granules. As a result, some local granules that define $Rain_L$ will overlap with target granules that define $Drizzle_T$. These ambiguous granules at the intersections of concepts will be of low quality, and contain misclassified records. This will result in low *Lg.Pr* and *Tg.Pr* values, giving these granules a low quality score. In order to successfully match $Rain_T$ to $Rain_L$ and $Drizzle_L$ to $Drizzle_T$ we must consider how many high-quality granules overlapped between $Rain_T$ and $Rain_L$ compared to low quality granules. In **Definition 9** we define the Match Score relation *MS* that represents the quality of a match between concepts based on the quality of granules used.

In addition to overlapped data-points, we must consider a concept's position in the ontology's hierarchy. Here we are concerned with matching concepts at the correct *ontology level* identified by $\lambda$. Consider matching the target $LiquidPrecipitation_T$ to the local $Precipitation_L$. We can infer that all granules that represent $Rain_T$ and $Drizzle_T$ will also represent their super-class $LiquidPrecipitation_T$. To see that this is the case, consider the fact that the rules for any super-class are inclusive of *all* data-points for all sub-classes, and their granules: so any data-points for $Rain_T$ and $Drizzle_T$ will also be the data-points for $LiquidPrecipitation_T$. However, this same fact cannot be used to match a local and target concept without considering at which $\lambda$ level the individual concepts reside.

At ontology $O_L$ level $\lambda = 3$ we have $Drizzle_L$, $Rain_L$, $Snow_L$ and $IcePelletes_L$ and their direct super-class $Precipitation_L$ at level $\lambda = 2$. In $O_T$ we also see $Precipitation_T$ at $\lambda=2$, but $Drizzle_T$, $Rain_T$, $Snow_T$ and $IcePelletes_T$ are at level $\lambda = 4$. Assuming their data-points overlap, we can match the concepts at level $\lambda_L = 3$ and $\lambda_T = 4$ as well as at levels $\lambda_L = 2$ and $\lambda_T = 2$ strictly by considering the granules. Matching $LiquidPrecipitation_T$ and $SolidPrecipitation_T$ at $\lambda_T = 3$ to their $O_L$ counterpart is more difficult because there is no direct match. In **Definition 10** we define the Level Match relation *LM* between a target concept *TC* and a set of local concepts $\{LC_i\}$ it matched at a particular local level $\lambda_L$.

The union of data-points for $LiquidPrecipitation_T$ and $SolidPrecipitation_T$ overlap with data-points for $Precipitation_L$. Equally so, $LiquidPrecipitation_T$ can be matched to the union of data-points for $Drizzle_L$ and $Rain_L$, and finally $SolidPrecipitation_T$ to the union of data-points for $Snow_L$ and $IcePellets_L$. In order to find the best match within the $O_L$ hierarchy, we attempt to match $LiquidPrecipitation_T$ to all local concepts who's data-points overlapped with its own data-points at different $\lambda_L$ levels of $O_L$. By doing so, $LiquidPrecipitation_T$ may get a partial match for $Rain_L$ and $Drizzle_L$ at $\lambda_L = 3$. It will also match a portion of the data-points for $Precipitation_L$ at $\lambda_L = 2$. Because a

direct match for $LiquidPrecipitaiton_T$ does not exist in $O_L$, a "best guess" is chosen instead with a lower match rating reflecting the indirect match. To distinguish between good and bad Level Matches at a particular local level $\lambda_L$, as defined in **Definition 10**, we create the Match Rank score *msv* in **Definition 11** that assigns a numeric rank to a particular Level Match.

To identify highly distinct data-point clusters within levels, we identify concepts whose matched granules are significantly better than others. For example, $Rain_T$ and $Drizzle_T$ may both overlap with $Rain_L$, although $Drizzle_L$ to a lesser degree. Also, $Snow_T$ may overlap with $Rain_L$, but to an even lesser degree: perhaps some weather stations recorded *rain* during a cold front, and classified it as *snow*, while capturing "rain like" data-points. To accommodate this use-case, the algorithm identifies *gaps* in the quality of matches. In the example of matching target concepts to $Rain_L$, there may be a slight gap between the ranking of the "$Rain_T$ to $Rain_L$" match, and the "$Drizzle_T$ to $Rain_L$" match, and a larger gap between these two and the "$Snow_T$ to $Rain_L$" match which would be very low on the ranking scale. In **Definition 12** we define the Gap Match ranking that implements this method in order to distinguish high ranking "close matches" from significantly lower ranking matches.

In the same manner as stated above, $Rain_T$ data-points will overlap with $Rain_L$, as well as the more inclusive $Precipitation_L$. In fact, due to ambiguous data-points in $Drizzle_L$, $Rain_T$ may have more data-points overlap with $Precipitation_L$ rather than $Rain_L$. In order to prevent such mismatches, our algorithm also considers the hierarchical position and quality of matched sibling concepts with target concepts. For example, if $Rain_T$ closely matches $Rain_L$ and $Precipitation_L$, we look at its sibling concept $Drizzle_T$ to determine a best match. In this case, $Drizzle_T$ has a match with $Drizzle_L$ and $Precipitation_L$. Here, $Drizzle_L$ is a local sibling of $Rain_L$. Taking this into consideration, we see a hierarchical pattern emerging, and it is this: the siblings $Rain_T$ and $Drizzle_T$ both match $Precipitation_L$; $Precipitation_L$ has the sub-classes $Rain_L$ and $Drizzle_L$; $Rain_T$ matched $Rain_L$ and $Drizzle_T$ matched $Drizzle_L$, both sibling pairs. The algorithm then infers two hypotheses from this information. The first hypothesis is that by having many sub-classes (in addition to overlapping data-points) in common, the two super-classes $Precipitation_L$ and $Precipitation_T$ are equivalent. This is defined in **Definition 13** as the Gap To Parent Match. The second hypothesis is that given many overlapping data-points and equivalent super-classes, $Rain_T$ is equivalent to $Rain_L$, and $Drizzle_T$ is equivalent to $Drizzle_L$. The second hypothesis is defined in **Definition 14** as a Child via Parent Match. This measure is also meant to identify matched concepts that may have many overlapping data-points, but who's siblings do not.

Depending on the application and domain of the ontologies being matched, a user may want to indicate a cut off point for the quality of matches she will consider valid. For this purpose we define the Close Match relation *CM* in **Definition 15** that identifies matches that met this cut off point. For further verification by the user, we group such "close matches" by the highest super-class found, as demonstrated in Appendix 5 on page 112. For example, based on the matching process, the following target concepts have been identified as a Close Match with $LiquidPrecipitation_T$: $Precipitation_L$ $Drizzle_L$, $Rain_L$. This will be displayed to the user as

$$\text{CMSubs}(\ \text{LiquidPrecipitation}_\text{T}\ ) = \{\ \text{Precipitation}_\text{L} \sqsupseteq \{\ \text{Drizzle}_\text{L},\ \text{Rain}_\text{L}\ \}\ \},$$

where CMSubs is the set of concepts matched to $LiquidPrecipitation_T$ called Close Match Sub-classes, as defined in **Definition 16**. This hierarchical perspective is meant to make it relatively easy for a user to pinpoint which of the three concepts ($Precipitation_L$, $Dirzzle_L$, or $Rain_L$) $Precipitation_T$ should match.

Depending on the user's point of reference, she may be interested in viewing the matches from either the local or target ontology's perspective. For this reason, in **Definition 17** we define the Local To Target Matches set L2TM and Target To Local Matches set T2LM. These are partially ordered sets that list matches sorted in such a way that the best matches are shown first. Both sets contain identical information, the only difference being that L2TM displays matched grouped by concepts in the local ontology, and T2LM by concepts in the target ontology.

In the following sections 3.11 and 3.12 we present the formal definitions of the terms just discussed.

## 3.11 Match Rankings

In order to identify the quality of a particular match between a local concept (*LC*) and a target concept (*TC*), the "Match Score" (MS) considers the quality of the measures used to create the match, as per **Definition 9**. Recall from **Definition 3** on page 36 that each granule implies a particular concept. For example, the local granule *Lg* implies the local concept *LC*, which is identified by $Lg \Rightarrow_G LC$. Similarly the target granule *Tg* implies the target concept *TC*, which is identified by $Tg \Rightarrow_G TC$. The match quality measure considered is the quality of the target granules used to create the match. Again recall from **Definition 4** on page 37, that each granule represents a leaf node in a decision tree, and that each granule has a probability associated with a leaf node. This probability, identified as *Tg.Pr*, is the number of *correctly* classified records in the leaf node divided by the number of *all* records classified by the leaf node. The derived numeric value *msv*, as per **Definition 9**, is the average quality

*Tg.Pr* of target granlues matched with local granlues for a partcicular *LC* and *TC* pair. This average considers the probability of the same *Tg* matched to multiple *Lg*'s as independent probabilities, as per the example in Table 7 below.

**Definition 9: (Match Score MS).** The Match Score is a ternary relation called *MS* between a local concept *LC*, a target concept *TC*, and a numeric value *msv* that represents the quality of the match between *LC* and *TC*. The numeric value *msv* in the relation *MS*(*LC*, *TC*, *msv*) utilises the probability of the target granules (*Tg*) matched to a local granule (*Lg*) in the relation *P2RM*(*Lg*, *Tg*) as per **Definition 5** and **Definition 6**. It is the average of he quality of target granules matched with a local granules. If a $Tg_j$ was matched to more than one $Lg_i$, that $Tg_j$'s quality is counted multiple times since it is associated with a different $Lg_i$:. The *msv* score is calculated by:

$$msv\ (\ LC, TC\ ) = \frac{\sum(Tg_j\ .\ Pr)}{|Lg_i|} \qquad (\ 6\ )$$

where:

$Lg_i$ ::= $Lg_i$ is a local granule associated with *LC*, where $Lg_i \Rightarrow_G LC$, matched with any target granule $Tg_j$.

$Tg_j$ ::= $Tg_j$ is a target granule associated with *TC*, where $Tg_j \Rightarrow_G TC$, matched with a local granule $Lg_i$.

for all existing relations *P2RM*($Lg_i$, $Tg_j$) that are members of the set $P2RM_{set}$ for the relation *ConceptMatch(LC*, *TC*, $P2RM_{set}$).

**Table 7. Match Score Example MS(*LC*, *TC*, *msv*)**

| **Local Granule** $Lg_i \Rightarrow_G LC$ | ***Target Granule*** $Tg_j \Rightarrow_G TC$ | ***Tg.Pr*** | |
|---|---|---|---|
| $Lg_0$ | $Tg_1$ | 0.9 | |
| $Lg_1$ | $Tg_1$ | 0.9 | |
| $Lg_2$ | $Tg_2$ | 0.95 | |
| $Lg_3$ | $Tg_2$ | 0.95 | |
| $Lg_6$ | $Tg_2$ | 0.95 | |
| $Lg_7$ | $Tg_5$ | 0.8 | |
| **Total** | **6** | **5.45** | ***msv* = 0.91** |

All *MS*(*LC, TC*, *msv*) relations are first grouped by the matched target concept *Tg*, and second by the local concept's hierarchical level as identified by $Level_L$ and defined in **Definition 1** on page 21. This

grouping will be used to compare matches between a target concept *TC* and local concepts at different levels of the local ontology $O_L$.

**Definition 10: (Level Match LM).** In order to compare matches between local concepts *LC* and target concepts *TC* in a more meaningful way, we group the matches at different levels of the local ontology. Specifically, *LM is a ternary relation between LC, TC, and* $Level_L(LC)=\lambda$ *as per* **Definition 1** *on page 21, and is identified by LM(LC, TC,* $\lambda$*).* We also define a partially ordered set LM(*TC,* $\lambda$) , whose members are all *LM* relations between a target concept *TC,* and all local concepts in $\{LC_i\}$ at $O_L$ level $\lambda$, sorted in descending order by *msv*.

To identify the best matches we make an assumption that highly scored matches will identify the best concept in a sub-class/super-class hierarchy in which the actual match exists. We describe the factors required to test this assumption in the rest of this chapter. In section 3.12 we discuss how those factors are utilized. In order to incorporate the hierarchical position of a local concept *LC* in the match ranking algorithm, we define *LM* which is a ternary relation between the local concept *LC*, the target concept *TC*, and the level at which *LC* is within $O_L$, as defined in **Definition 10**. For each *TC* and $O_L$ level $\lambda$, all such *LM*(*LC*, *TC*, $\lambda$) relations are contained in the partially ordered set LM(TC, $\lambda$), in descending order by *msv*, as per **Definition 10**.

**Definition 11: (Match Rank MRank).** The Match Rank is meant to assign a score to *LM* relations between a target concept *TC* and all local concepts *LC* at the same $O_L$ level for which a *LM*(*LC*, *TC*, $Level_L(LC)$) relation exist. It utilizes the *msv* score in the relation *MS*(*LC*, *TC*, *msv*) as per **Definition 9**, to calculate this ranking. Each Match Rank is sorted by the associated *msv* score, with highest score having the highest rank value. Specifically, *MRank is a ternary relation between a target concept TC, a local concept* $LC_x$ *in* $\{LC_k\}$*, and mr, referred to as* $MRank(TC, LC_k, mr)$*. Here, the partially ordered set* $\{LC_k\}$ *ordered on mr contains all the local concepts* $LC_x$ *that are at* $O_L$ *level* $\lambda$*, and have a LM relation with TC in the set LM(TC,* $\lambda$*). mr is a numeric rank normalized over [ 0, 1 ] defined by Equation 7 below.* Each *MRank* relation is also a member of the partially ordered set $MRank_{set}$, sorted in descending order by *mr*.

$$mr = 1 - \frac{k}{m} \qquad (7)$$

where *m* is the number of local concepts at a particular $O_L$ level $\lambda$ defined by $Level_L(LC_k)$, *k* is the index of $LC_k$ within $\{LC_k\}$ and $k \in \{1, \ldots, m\}$

To numerically determine which match is better, we introduce the *MRank* relation, and the associated numerical rank *mr* in **Definition 11**. The *mr* rank will determine which *LM* relation is better within the set LM(TC, $\lambda$), at a particular $O_L$ level $\lambda$, for a particular target concept *TC*. Continuing the example of matching $Precipitation_L$ and $Precipitaiton_T$, *mr* at level 3 may determine that $LM(Rain_L, Rain_T, 3)$ is ranked higher than $LM(Drizzle_L, Rain_T, 3)$, stating that $Rain_T$ and $Rain_L$ are a better match. $Precipitation_L$ is the only concept at $O_L$ level 2 matched to $Rain_T$, so $LM(Precipitation_L, Rain_T, 2)$ is considered best at level 2. However, the local concepts $Rain_L$ and $Drizzle_L$, will also be matched to $Rain_T$'s super-class $LiquidPrecipitation_T$, because of its large and inclusive cluster, as discussed in section 3.10. The next section describes how matches between multiple target concepts and multiple local concepts are ranked, with the use of "matching schemes".

## 3.12 Matching Schemes

Once the initial matches between a target concept *TC* and multiple local concepts $\{LC_x\}$ are determined, the matches are rated based on additional matching criteria called *matching schemes*. Matching schemes focus on structural similarities between local and target concepts, while utilizing the *MRank* relation and associated *mr* rank between those concepts, as defined in **Definition 11**. Matching schemes begin by considering the *mr* value of each *MRank* relation in the $MRank_{set}$ set, at some $O_L$ level $\lambda$, and predict additional matches based on structurally relevant similarities. These similarities build on top of the *anchors* built by previous matching processes, and act as prerequisites for further matching schemes.

**Definition 12: (Gap Match GM).** The Gap Match represents the best matches between local concepts at some $O_L$ level $\lambda$ and a target concept *TC*. It identifies a *significant* gap between the *mr* score of *MRank* relations between *TC* and those local concepts. Specifically, *GM is a binary relation* $GM(LC_k, TC)$ *between a local concept* $LC_x$ *and a target concept TC, where the* $mr_k$ *rank in the relation* $MRank(TC, LC_x, mr_x)$ *is significantly larger than* $mr_m$ *for the next relation* $MRank(TC, LC_m, mr_m)$ *in the partially ordered set* $MRank_{set}$. *Here we consider all MRank relations in* $MRank_{set}$ *for a single TC and a single local ontology level* $\lambda$, *where* $\lambda = Level_L(LC_k) = Level_L(LC_m)$. *A significant gap occurs when* $(mr_k - mr_m) \geq GAP\_MIN$, *where GAP_MIN is a user defined threshold*. In order to omit large gaps between low ranking *MRank* relations, only relations with $mr \geq$ GAP_CHECK_MIN are considered, where GAP_CHECK_MIN is a user defined threshold. Each *GM* relation is a member of the set $GM_{set}$.

The first scheme is the Gap Match scheme identified by the *GM* relation, as per **Definition 12**. The corresponding $GM_{set}$ is a set containing all *GM* relations, and represents any local to target concept matches that have a large gap between the *mr* rank of two consecutive *MRank* relations, within $MRank_{set}$. For example, consider

$MRank_{set}$ = { $MRank_0$ ( $LiquidPrecipitation_T$, $Rain_L$, 0.9 ),
$MRank_1$ ( $LiquidPrecipitation_T$, $Drizzle_L$, 0.9 ),
$MRank_2$ ( $LiquidPrecipitation_T$, $Snow_T$, 0.1 ) }.

$MRank_0$ and $MRank_1$ relations are equally distinct from the $MRank_2$ relation. Here we see a significant gap between $mr_0 = mr_1 = 0.9$ and $mr_2 = 0.1$. Such gaps occur when the decision trees made to differentiate local siblings such as $\{Rain_L, Drizzle_L, Snow_L\}$ are similar to the decision trees made to differentiate target siblings. In order to use only gaps between high ranking *MRank* relations, only *MRank* relations with $rm \geq$ GAP_CHECK_MIN are considered. Here, the user defined numeric threshold GAP_MIN identifies what value constitutes a significant gap. Also, if consecutive *MRank* relations have the same *mr* value, such as $mr_0 = mr_1 = 0.9$, they are compared as a single *mr* value and *MRank* to the next *MRank* relation with a different *mr* value , like $mr_2 = 0.1$. As a result only the binary relations $GM(LiquidPrecipitation_T, Rain_L)$ and $GM(LiquidPrecipitation_T, Drizzle_L)$ become members of the set $GM_{set}$, as per **Definition 12**.

**Definition 13: (Gap To Parent Match G2PM).** The Gap To Parent Match represents a match between a local concept and a target concept, based on the number of matches between their direct sub-classes. Specifically, *if for some local concepts* $LC_x$ *and* $LC_m$ *, where* $LC_x \sqsubseteq_{direct} LC_m$ *, and target concepts* $TC_y$ *and* $TC_n$ *, where* $TC_y \sqsubseteq_{direct} TC_n$ *, the relations* $GM(LC_x, TC_y)$ *and* $GM(LC_m, TC_n)$ *are members of* $GM_{set}$ *, the relation* $MRank(LC_m, TC_n, mr)$ *is a member of* $MRank_{set}$ *with* $mr \geq$ *TOP_P2P_RANK_MIN, then the relation* $G2PM(LC_m, TC_n, count)$ *is a member of* $G2PM_{set}$. Here, *count* is the number of *GM* relations between sub-classes of $LC_m$ and $TC_n$, and TOP_P2P_RANK_MIN is a user defined threshold identifying the minimum value of an *rm* rank in a *MRank* relation between a local and target concept before those two concepts can qualify for the *G2PM* relation. Each *G2PM* relation is a member of the set $G2PM_{set}$.

Next, the "Gap To Parent Match" scheme is identified by the relation *G2PM*, as per **Definition 13**. This scheme creates a *G2PM* relation between a local and a target concept, based on a match between these two concepts, as well as a match between their direct sub-classes. Essentially, if a subset of local sibling concepts $\{Rain_L, Drizzle_L\}$ match a subset of target sibling concepts $\{Rain_T, Drizzle_T\}$, and their direct super-classes $LiquidPrecipitation_T$ and $Precipitaiton_L$ are matched, this is a good indicator

that these super-class concepts are a *good* match. The first requirement for the algorithm to categorize such matches as *good* is that the *GM* relations between the sibling classes be members of $GM_{set}$. This means that both $GM(Rain_L, Rain_T)$ and $GM(Drizzle_L, Drizzle_T)$ must be members of $GM_{set}$. The second requirement is that the *GM* relation between the parent concepts $GM(Precipitation_L, LiquidPrecipitation_T)$ also be a member of $GM_{set}$.

The third requirement for the "Gap To Parent Match" between a local and target concepts is that the $GM(Precipitation_L, LiquidPrecipitation_T)$ relation be of *high quality*. For this we consider the Match Rank defined in **Definition 11**. If such a relation is a member of $GM_{set}$, then the relation $MRank(Precipitation_L, LiquidPrecipitation_T, mr)$ is a member of $MRank_{set}$, as per **Definition 11** and **Definition 12**. We now utilize the *rm* rank in their *MRank* relation as an indicator for the quality of the match. If this *rm* rank is greater or equal to a user defined threshold TOP_P2P_RANK_MIN, then a match between $Precipitation_L$ and $LiquidPrecipitation_T$ is considered of *high quality*. Setting a higher TOP_P2P_RANK_MIN threshold means a user put great emphasis on the role sibling concepts play in the matching of their super-classes.

As a final measure of quality for the match between local and target concepts $Precipitation_L$ and $LiquidPrecipitation_T$, we also consider the number of matches between their direct sub-classes. To capture this we create the *count* value, which is the number of *GM* relations between their sub-classes. Finally, if all conditions defined in **Definition 13** are met, we create the *G2PM*(*LC*, *TC*, *count*) relation, which becomes a member of the set $G2PM_{set}$. In our example, having created the $GM(Rain_L, Rain_T)$ and $GM(Drizzle_L, Drizzle_T)$ relations, we would also have the $G2PM(Precipitation_L, LiqiudPrecipitation_T, 2)$ relation.

**Definition 14: (Child via Parent Match CVPM).** The "Child via Parent Match" is a relation between a local concept *LC* and a target concept *TC*, based on two criteria: a successful granule overlap for the two concepts (*ConceptMatch* as per **Definition 6**), and a successful "Gap To Parent Match" between their direct parent classes (*G2PM* as per **Definition 13**). Specifically, *CVPM is a binary relation between a pair of concepts LC and TC, if 1) the relation ConceptMatch(LC, TC,* $P2RM_{set}$*) is an element of* $CM2C_{set}$ *; and 2) the relation* $G2PM(LC_m, TC_n, count)$ *between their direct super-classes* $LC_m$ *and* $TC_n$*, is an element of* $G2PM_{set}$*., where* $LC \sqsubseteq_{direct} LC_m$ and $TC \sqsubseteq_{direct} TC_n$. All *CVPM* relations are elements of the set $CVPM_{set}$.

The third matching scheme is the "Child via Parent Match" identified by the relation *CVPM*, as defined in **Definition 14**. This relation identifies a match between a local concept *LC* and a target concept *TC*, based on their individual concept definitions and associated granules, as well as a *significant* match between their super-classes. Recall that the "Gap To Parent Match" in **Definition 13** identifies a match between two concepts if their direct sub-classes were matched. This is a bottom-up process in that the super-classes are matched because their direct sub-classes were matched. In contrast, the "Child via Parent Match" is a top-down process in that the direct sub-classes are matched because their direct super-classes were matched. The reason for this matching scheme is that two concepts *LC* and *TC* matched solely by their granules, meaning data-points overlapping with rules, may not have a significantly high "Match Rank" as per **Definition 11**. In such a case, we look to their siblings and super-classes for more information.

Consider again the local and target *Precipitation* concept hierarchies, and specifically the bottom local and target concepts $IcePellets_L$ and $IcePellets_T$. We may already have a good quality match between the local and target *Rain*, Drizzle, and *Snow* concepts, and a "Gap To Parent Match" between $Precipitation_L$ for both $LiquidPrecipitaiton_T$ and $SolidPrecipitaiton_T$. We now want to use that information to see if we can match $IcePellets_L$ to $IcePellets_T$. Unfortunately we cannot use these concepts in isolation because they have a low quality match, perhaps due to lack of weather station records for $IcePellets_T$. We can see that 1) $IcePellets_L$`s siblings were all matched to each other, and that 2) their super-classes were matched as well. We can also see that 3) some $IcePellets_L$ data-points did overlap with $IcePellets_T$ data-points, although the data-points belong to low quality granules. We use these 3 observations to infer that $IcePellets_L$ and $IcePellets_T$ are in fact equivalent concepts, and build the binary relation $CVPM(IcePellets_L, IcePellets_T)$ as defined in **Definition 14**.

**Definition 15: (Close Match CM).** A match between a local and a target concept is considered a Close Match when the *rank* of the match is above a predetermined threshold. Specifically, *CM is a binary relation between a local concept LC, a target concept TC, represented as CM(LC, TC) if the relation MRank(LC, TC, mr) is a member of* $MRank_{set}$*, and* $mr \geq TOP\_RANK\_MIN$, where TOP_RANK_MIN is a user defined constant. All *CM* relations are also part of the set $CM_{set}$.

**Definition 16: (Close Match Sub-classes CMSub).** For verification purposes by the user, all *CM* relations between a target concept *TC* and local concepts in the same local sub-tree are members of a partially ordered set $CMSub_{TC}$, sorted on the $O_L$ level, $Level_L(LC_x)$, of each local concept $LC_x$. Specifically, $CMSub_{TC}$ *is a partially ordered set whose members are CM relations in* $CM_{set}$ *between a*

*particular target concept TC and local concepts {$LC_x$}, where each $LC_x$ in {$LC_x$} is a sub-class of LC, and CM(LC, TC) is also in $CM_{set}$.* Each CMSub(*TC*) set is a member of the collection $\mathfrak{CM}$.

The fourth and final matching scheme is "Close Match", and is identified by the relation *CM*, as per **Definition 15**. This relation identifies highly ranked matches between a local (*LC*) and target (*TC*) concept, and is based on their *mr* rank in the *MRank*(*LC*, *TC*, *mr*) relation, if one exists. For a match to be considered a "Close Match" the associated *mr* rank must be greater or equal to a user defined threshold called TOP_RANK_MIN. This threshold allows a user to state how strict the matching process is, by only including high ranking matches in the $CM_{set}$ set. As a supportive measure in manual verification of matches, we define the partially ordered set CMSub(*TC*), as defined in **Definition 16**. Each CMSub(*TC*) set allows a user to quickly see matches from a hierarchical perspective, and remove possible false positives between a target concept *TC* and sub-trees of local concepts. Each set CMSub(*TC*) is also a member of the collection $\mathfrak{CM}$, which is ultimately displayed to the user.

Revisiting the example from section 3.10, we may have Close Match relations with $LiquidPrecipitation_T$, mainly $Precipitation_L$ $Drizzle_L$, $Rain_L$. This will be displayed to the user as

$$\text{CMSubs}(\text{LiquidPrecipitation}_T) = \{\ \text{Precipitation}_L \sqsupseteq \{\text{Drizzle}_L, \text{Rain}_L\}\ \},$$

where CMSubs is the set of concepts matched to $LiquidPrecipitation_T$. By providing this hierarchical view, it should be relatively easy for a user to pinpoint which of the three concepts $Precipitation_L$, $Dirzzle_L$ or $Rain_L$, $LiquidPrecipitation_T$ should match.

**Definition 17: (Final Matches L2TM and T2LM).** For display purposes and verification by the user, all matches are grouped under either the local concepts and local ontology $O_L$ levels, or the target concept and target ontology $O_T$ levels. The matches are members of the matching scheme sets $GP_{set}$, $G2PM_{set}$, $CVPM_{set}$ and $CM_{set}$, with the associated confidence rating $ms_{rating}$, as listed in Table 8. Specifically, *L2TM is a partially ordered set that contains relations in each matching scheme set between local concepts and target concepts, sorted on the $O_L$ level and $ms_{rating}$ of the matching scheme used. T2LM is a partially ordered set that contains matches between target concepts and local concepts, sorted on the $O_T$ level and $ms_{rating}$ of the matching scheme used.*

**Table 8. Matching Scheme Rating ($ms_{rating}$)**

| Matching Scheme | $ms_{rating}$ | Description |
|---|---|---|
| Gap Match (*GM*) | 1.0 | Matching concepts based on a highly distinct Match Rank in *MRank*. |
| Gap To Parent Match (*G2PM*) | 0.75 | Matching two parent concepts based first on their children's GP match, and secondly on the closeness of their own *MRank*. |
| Close Match (*CM*) | 0.5 | A match between concepts based on their high *MRank*. |
| Child via Parent Match (*CVPM*) | 0.25 | A recheck of the original matches, i.e. *ConceptMatch* relations in $C2CM_{set}$, based on newly found "Gap To Parent Matches", i.e. *G2PM* relations in $G2PM_{set}$. |

Once the various matching schemes have been applied to local and target concepts, each produces sets of the relations, mainly $GM_{set}$, $G2PM_{set}$, $CM_{set}$ with the associated $\mathfrak{CM}$ collection, and the $CVPM_{set}$ set. Each matching scheme, and as a result each set, is ranked and combined into two sets that group "Local To Target Matches" ($L2TM_{set}$), and "Target To Local Matches" ($T2LM_{set}$), as defined in **Definition 17**, which are presented to the user for verification. The matches in the $L2TM_{set}$ set are grouped by local concepts, and this set is useful for users who are familiar with the local ontology $O_L$. Similarly, the matches in $T2LM_{set}$ are grouped by target concepts and this set is useful for users who are familiar with the target ontology $O_T$. In each set, the top level super-class concepts and highest rated matching schemes are displayed first. Both lists are equivalent, and differ only in the main grouping, where the primary concepts are either the local or target concepts. An example is created in Chapter 4, based on the Weather test case chosen for this thesis.

# Chapter 4. Results and Analysis

## 4.1 Evaluation Measures

The evaluation measures described in the following sections are meant to identify the *precision* and *recall* of the matching algorithm presented in this thesis. We adopt the notion of *correspondence proximity* proposed by Ehrig and Jérome [39], and extend it with our own proximity measures specifically created to measure the performance of our bottom-up matching algorithm. We call our approach a bottom-up matching algorithm because it associates ground terms with records in a database, and finds additional matches moving "upwards" towards the root of the ontology.

The evaluation methods used to validate the ontology matching algorithm in Chapter 4 are based on methods described by Shvaiko et. al. [72], in which ontology matching benchmarks are described and past competitions evaluated. The evaluation measures utilized by Shvaiko et. al., and adopted here are based on the work proposed by Ehrig and Jérome [39].

In 2006, Shvaiko et. al. introduce the Ontology Matching (OM) Workshop[41] and the Ontology Alignment Evaluation Initiative[42] (OAEI), and analyze the 2006 results. Although OM and OAEI have a vast amount of resources for ontology matching, including ontologies and alignments for verification, none of them have a corresponding dataset of records to use for data-mining, and for building decision trees required by this thesis. As a result, an existing weather ontology [73] and dataset [74] were manually associated, and used in the tests in the following chapter. The dataset is a collection of weather data from weather stations around the world in the METAR format, a format of meteorological codes defined by the World Meteorological Organization (WMO)[43] [75] to record meteorological phenomenon. The weather ontology is a modified version of an ontology by Elkiss et al. [73], which was originally used to convert METAR reports to DAML [76]. Without a subject matter expert in the meteorological domain, only one ontology was associated with multiple files in the METAR dataset format, as per the procedure described in section 3.3. As a real-life example test, the taxonomies and data records of two major consumer electronics retailers were extracted from their respective online catalogs, and matched using the algorithm. The "Simple Tests (1xx)" and "Systematic Tests (2xx)" tests in the Benchmark Track [72] are performed using the weather ontologies in order to analyze how a matching algorithm performs when matching ontologies with

[41] http://www.om2006.ontologymatching.org/
[42] http://oaei.ontologymatching.org/
[43] http://www.wmo.int

particular types of differences between them. The "Real-Life Test (3xx)" is performed on the electronics retailers' extracted taxonomies and data records.

The first test performed is the "Simple Tests (1xx)", which matches an ontology to itself, an irrelevant ontology, or scaled down version of the original in a less expressive representation in OWL-Lite. As a benchmark for our tests, we are matching a single ontology to itself using the same METAR data. We also match the same ontology to itself, but with a subset of the METAR data focusing on a specific geographical region dominated by a particular type of weather, such as arctic weather or sunny weather. The second test is the "Systematic Tests (2xx)", which matches an ontology to a modified version of itself, where some features are disregarded, and is meant to identify how the algorithm performs with variety of differences in target ontologies. These differences can be structural, lexical, or simply have missing or new elements in the target ontology. The modifications considered in the original OM test include (a) modifying labels of ontology entities; (b) modifying comments; (c) suppressing, expanding or flattening specialization hierarchies; (d) removing $\mathcal{ABox}$ instances; (e) removing attributes or modifying restrictions placed on their domain and range; (f) and expanding or flattening concepts [72].

This algorithm purposefully avoids using lexical information from any ontology elements, including labels or comments so the (a) and (b) tests are not performed. For generality of interpretations between a local and target ontology, any $\mathcal{ABox}$ instances provided with an ontology are ignored and complete focus is placed on database records, which means test (d) is not performed. It is important to note that while an $\mathcal{ABox}$ may contain instances that can be treated as database records, it can simply contain a small set of *examples*, not suitable for data-mining. For this reason, the algorithm requires an associated database or a great number of $\mathcal{ABox}$ instances suitable for data-mining. Since this algorithm is driven by the data, removing any $\mathcal{ABox}$ instances equates to the "Simple Tests (1xx)" test where instances are included only for a particular geographical region that focuses on a particular type of weather, as discussed in the previous paragraph.

Lastly, the only attributes that are referenced by the algorithm are ones that exist in the dataset associated with the ontology. As a result, each property that is not needed to represent a column in the dataset is ignored. Since test (e) deals specifically with how property removal affects the matching process, the test is not performed. Consequently, a test can be performed by removing several 2-dimensional trees (using 2 attributes at a time), in order to see how the matching algorithm performs without the attributes used to create the removed tree. Consequently, by modifying the structural

characteristics of the weather ontology, only tests (c) and (f) are performed. The third and final test is the "Real-life Tests (3xx)" which involves running a matching algorithm on a real life ontology of bibliographic information. Since a dataset was not included with the benchmark ontology to perform data-mining on, this test was performed using taxonomies and data records extracted from online catalogs for two consumer electronic retailers, *FutureShop* and *BestBuy*.

## 4.2 Evaluation Measures - Precision and Recall

It has been well established [77] that a solution to a search problem needs a good balance between identifying correct matches (true positives) and limiting the number of proposed matches to reduce incorrect matches (false positives). The tests described by Shvaiko et. al. [72] measure the *precision* and *recall* of the matching algorithm, and are based on the work of Ehrig and Jérome [39]. These precision and recall measures are also adopted here, with variations that specifically test the precision and recall of the matching algorithm proposed by this thesis. The details of the work done by Ehrig and Jérome are not discussed here, as evaluation methods in general are outside the scope of this thesis. Only the background information necessary to understand the motivation of the measures adopted here is provided, along with their definitions. *Precision* and *recall* is a classic measure that evaluates how good an algorithm is at identifying the correct matches [39]. *Precision* is the ratio between true positives and all potential matches retrieved. *Recall* is the ratio between true positives and all possible matches. This is summarized in **Definition 18** below.

**Definition 18:** (Precision and Recall). *Precision* is given by

$$Prec(M,R) = \frac{|R \cap M|}{|M|} \quad (8)$$

and *Recall* by

$$Rec(M,R) = \frac{|R \cap M|}{|R|} \quad (9)$$

where $M$ are all proposed matches, $r_{pair}(LC, TC)$, and $R$ are all the correct matches, usually as a predefined list used for verification.

There is also a need to generalize precision and recall, giving the generalized functions $Prec_{\omega}(M,R)$ and $Rec_{\omega}(M,R)$, respectively, to accommodate variation in degrees of correctness. This is allowed as long as the generalization methods adhere to a set of basic properties [39]. These properties includes

the fact that the generalized version of $|R \cap M|$, mainly $\omega(R, M)$, should not produce a negative number, and should not have a cardinality greater than $|R \cap M|$. Also a relaxed measure should add flexibility to Prec and Rec, producing better results. And finally, $Prec_\omega$ and $Rec_\omega$ should be symmetric where $\omega(R, M) = \omega(M, R)$. For ontology matching however, where the local and target ontologies are not necessarily symmetric, there is no guarantee that their predefined correct matches $R$ and proposed matches $M$ will be symmetric. For this reason the symmetry is not required here [39]. To calculate a relaxed measure, a "correspondence proximity" must be calculated which is the product of several match features.

**Definition 19: (Ground Term Precision).** *Ground Term Precision* is similar to the Precision in **Definition 18**, except it considers only matches to local ground terms.

As mentioned previously, the ontology matching algorithm presented here is a bottom-up approach, where matches between ground terms are assumed to be of the highest quality. This is due to their close association with database records, and the assumed consistency between local and target databases, as described in section 3.3. In order to test this theory, the *Ground Term Precision* measure is also calculated, as defined in **Definition 19**. It is similar to the *Precision* measure defined in **Definition 18**, except it only considers ground terms in its rankings.

**Definition 20: (Match Correspondence Proximity MCP).** MCP is a rating of a particular match, in the range of [ 0, 1 ], which considers several characteristics of the matching process and the matches themselves, giving a relaxed proximity measure.

$$\mathrm{MCP}(LC, TC) = CP_{pair}(LC, TC) \times CP_{rel}(LC, TC) \times CP_{conf}(LC, TC) \qquad (10)$$

where the characteristics are

$CP_{pair}(LC, TC)$ ::= $CP_{pair}(LC, TC)$ is the rating of proposed match between $LC$ and $TC$ in the [ 0, 1 ] range.

$CP_{rel}(LC, TC)$ ::= $CP_{rel}(LC, TC)$ is the rating of the relation between $LC$ and $TC$ in the [ 0, 1 ] range.

$CP_{conf}(LC, TC)$ ::= $CP_{conf}(LC, TC)$ is the confidence rating of the match between $LC$ and $TC$ in the [ 0, 1 ] range.

**Definition 21: (Full Match Correspondence Proximity).** The generalized accumulated proximity measure rates the matching algorithm itself, and is based on all matches *ms* produced by all matching schemes [39].

$$MCP_{full}(M,R) = \frac{\sum \mathrm{MCP}(\, ms(LS,\ TC),\ r(LC,\ TC)}{|M|} \quad (11)$$

and $M$ are all the proposed matches, $ms(\, LC\,,\, TC\,) \in M$, $ms$ is a Match Scheme, and $R$ are the predefined correct matches where $r(\, LC, TC\,) \in R$.

The Match Correspondence Proximity MCP is adopted from "correspondence proximity" proposed by Shvaiko et. al. [72] and defined in **Definition 20**. MCP represents a rating of the match between *LC* and *TC*, and considers several elements of the matching process. It incorporates the characteristics of the method used to create the original match as well as the similarities and differences between *LC* and *TC*. Because this measure rates the matching algorithm itself, it measures the proposed matches *M* against the actual matches *R*. Note that each characteristic of MCP, mainly $CP_{pair}$, $CP_{rel}$, and $CP_{conf}$, is aggregated through MCP to the final rating of the ontology matching algorithm $\mathrm{MCP}_{\mathrm{full}}$, as defined by **Definition 21**. This means that these methods must also abide by the requirement of being in the range of [ 0, 1 ].

**Definition 22: (Effort and Normalization).** The effort to go from concept $LC_A$ to concept $LC_B$ in a single branch of a hierarchy is the edge count between these two concepts. In order to normalize this value to the range of [ 0, 1 ], the number of edges is divided by the depth of their branch.

$$effort(LC_A, LC_B) = |Level_L(\, LC_A) - Level_L(\, LC_B)| \quad (12)$$

$$effort_{norm}(LC_A, LC_B) = 1 - \left(\frac{effort(\, LC_A, LC_B)}{Level_L(LC_C)}\right) \quad (13)$$

where either $LC_C \sqsubseteq LC_B \sqsubseteq LC_A$ or $LC_C \sqsubseteq LC_A \sqsubseteq LC_B$, and $LC_C$ is a ground term.

The first measure to influence MCP is $CP_{pair}$, the rating of the proposed match $ms(LC\,,\, TC)$ and the correct match $r(LC, TC)$. Note that this is not rating the different matching schemes, only the type of match a scheme produced. For example, consider $ms(Precipitation_L\ ,\ Rain_T)$ which matched $Precipitaton_L$ with its direct sub-class, where $Rain_L \sqsubseteq_{\mathrm{direct}} Precipitation_L$ and $r_{pair}(Rain_L, Rain_T)$. A match between a concept and its sub-class is relatively good, especially considering it is matched to its *direct* sub-class. This type of match would be rated as *better* when compared to a match between a concept and its *indirect* sub-class, because the distance in terms of edge counting between the two

concepts is larger. *Edge counting* is a well established method of determining the distance between two nodes in a graph, and has been successfully applied to ontology matching [12]. The match would also be rated as *better* when compared to a match between a concept and its super-class. This is because a match to a sub-class is more specific than a match to a super-class [39]. To demonstrate this point, consider that every concept could be matched to the root concept, but such a match should be scored relatively low as it is far too ambiguous to be of any benefit. A direct match would have the highest rating of 1.0.

The second measure influencing MCP is $CP_{rel}$, which is a rating of the *type of relationship* between *LC* and its match. Since the use cases evaluated in the next chapter deal strictly with hierarchical ontologies which only have sub-class and super-class relationships, $CP_{rel}$ is by default set to 1.0, or $effort_{norm}$, a rating based on edge counting, as defined in **Definition 22**.

The final measure influencing MCP is $CP_{conf}$, which is the confidence given to the matching technique, either based on the scheme's $ms_{rating}$ rating (listed in Table 8) which produced the match, or the proximity measure's own rating. These values are given in the following section.

## 4.3 Proximity Measures

The proximity measures described in this section are *concrete* proximity measures based on the general definition of Match Correspondence Proximity in **Definition 20**. As mentioned previously, the $CP_{rel}$ rating is by default 1.0, because we only consider super-class and sub-class relationships in these tests. This changes when we consider the hierarchical difference of a concept and its super-class or sub-class. In Table 9 below, we identify the types of *correct* relationships that can exist between concepts in the ontology. These are then used to identify how to rate a proposed direct match in *M* to its correct counterpart in *R*, if one exists, using the associated $CP_{pair}$, $CP_{rel}$, and $CP_{conf}$ ratings for the individual proximity measures in Table 10 to Table 15.

**Table 9. Proximity measure relationships**

| Found Relation | Correct Relation | *Definition* |
|---|---|---|
| $LC \equiv TC$ | $LC \equiv TC$ | A direct match between *LC* and *TC* |
| $LC \equiv TC$ | $LC \Uparrow TC$ | *LC* and *TC* are sibling, i.e. they share a *direct* super-class |
| $LC \equiv TC$ | $LC \sqsubseteq TC$ | *LC* was matched to its super-class *TC* |
| $LC \equiv TC$ | $LC \sqsupseteq TC$ | *LC* was matched to its sub-class *TC* |

The first concrete proximity measure is the *standard* proximity measure. It considers only direct matches as correct, and all other indirect matches as incorrect. This is achieved by assigning $CP_{pair}$ a value of 1.0 for all direct matches, and a value of 0.0 for any indirect matches, as per Table 10.

**Table 10. Standard proximity measure**

| Found Relation | Correct Relation | $CP_{pair}$ | $CP_{rel}$ | $CP_{conf}$ | *Comment* |
|---|---|---|---|---|---|
| $LC \equiv TC$ | $LC \equiv TC$ | 1.0 | 1.0 | 1.0 | Only considering direct matches. |
| $LC \equiv TC$ | $LC \Uparrow TC$ | 0.0 | 1.0 | 1.0 | Not considering indirect matches |
| $LC \equiv TC$ | $LC \sqsubseteq TC$ | 0.0 | 1.0 | 1.0 | Not considering indirect matches |
| $LC \equiv TC$ | $LC \sqsupseteq TC$ | 0.0 | 1.0 | 1.0 | Not considering indirect matches |

In order to rate the *Matching Scheme* used to perform a particular match, the *Match Scheme* proximity measure is configured with the MCP parameters in Table 11. Any match, direct or indirect is considered valid, with a rating of 1.0 assigned to $CP_{pair}$. In order to incorporate the *Matching Scheme*, the $CP_{conf}$ value is assigned the ranking of the particular scheme used, as defined in Table 8.

**Table 11. Matching Scheme proximity measure**

| Found Relation | Correct Relation | $CP_{pair}$ | $CP_{rel}$ | $CP_{conf}$ | *Comment* |
|---|---|---|---|---|---|
| $LC \equiv TC$ | $LC \equiv TC$ | 1.0 | 1.0 | $ms_{rating}$ | A direct match has the highest proximity |
| $LC \equiv TC$ | $LC \Uparrow TC$ | 1.0 | 1.0 | $ms_{rating}$ | Matched sibling have the second highest proximity |
| $LC \equiv TC$ | $LC \sqsubseteq TC$ | 1.0 | 1.0 | $ms_{rating}$ | A match within own hierarchical branch |
| $LC \equiv TC$ | $LC \sqsupseteq TC$ | 1.0 | 1.0 | $ms_{rating}$ | A match within own hierarchical branch |

To add flexibility to proximity measures, we now define measures that consider indirect matches as well as direct ones. The *symmetric* proximity measure identifies structural symmetry between matched concepts, by rating indirect matches with a value greater than 0.0. As shown in Table 12, the *symmetric* measure assigns different values to the $CP_{pair}$ rating, based on the type of *correct* relationship the matched concepts have. A correctly identified direct match has a rating of 1.0. Indirect matches are divided into two groups, siblings with a rating of 0.75, and super/sub-classes with a rating of 0.5. The matching scheme is not considered, as indicated by the assignment of 1.0 to $CP_{conf}$.

**Table 12. Symmetric proximity measure**

| Found Relation | Correct Relation | $CP_{pair}$ | $CP_{rel}$ | $CP_{conf}$ | *Comment* |
|---|---|---|---|---|---|
| $LC \equiv TC$ | $LC \equiv TC$ | 1.0 | 1.0 | 1.0 | A direct match has the highest proximity |
| $LC \equiv TC$ | $LC \Uparrow TC$ | 0.75 | 1.0 | 1.0 | Matched sibling have the second highest proximity |
| $LC \equiv TC$ | $LC \sqsubseteq TC$ | 0.5 | 1.0 | 1.0 | A match within own hierarchical branch |
| $LC \equiv TC$ | $LC \sqsupseteq TC$ | 0.5 | 1.0 | 1.0 | A match within own hierarchical branch |

To further differentiate between indirect matches, the *effort* proximity measure favours matches between a local concept and its target own sub-class. Generally speaking, sub-classes are viewed as a specialized version of their super-class [39]**.** The criteria which was used to create the match is then viewed as a specialization of the matched concept. Conversely, a match between a concept and its super-class is viewed as a generalization of that concept. As a result, the *effort* measure assigns a higher rating for matches between a concept and its sub-class, and a lower rating for a match with its super-class.

As indicated in Table 13, the $CP_{pair}$ is again used to differentiate between different types of matches, assigning 1.0 for direct matches, 0.8 to siblings, 0.4 to super-class matches, and 0.6 to sub-class matches. The word "effort" is adopted from Ehrig and Jérome [39], and refers to the amount of *effort* a user verifying the matches would need to manually correct or find the proposed matches. Comparing the *effort* measure to the *symmetric* measure in Table 12, the key difference is that the *effort* measure differentiates between super-class and sub-class matches. This is because symmetry does not consider the specialization and generalization mentioned above which is introduced with sub-classes and super-classes, respectively. However, Ehrig and Jérome do not consider the *distance* between the concept and its matched counterpart in the hierarchy. The next proximity measure introduced here does just that.

**Table 13. Effort proximity measure**

| Found Relation | Correct Relation | $CP_{pair}$ | $CP_{rel}$ | $CP_{conf}$ | *Comment* |
|---|---|---|---|---|---|
| $LC \equiv TC$ | $LC \equiv TC$ | 1.0 | 1.0 | 1.0 | A direct match has the highest proximity |
| $LC \equiv TC$ | $LC \Uparrow TC$ | 0.8 | 1.0 | 1.0 | Matched siblings have the second highest proximity |
| $LC \equiv TC$ | $LC \sqsubseteq TC$ | 0.4 | 1.0 | 1.0 | Matching to a more ambiguous super-class |
| $LC \equiv TC$ | $LC \sqsupseteq TC$ | 0.6 | 1.0 | 1.0 | Matching to a more specific sub-class |

The *effort with edge count* proximity measure is identical to the *effort* measure, except that it discriminates against large hierarchical differences between matches. This measure uses the number

of edges between a concept and its matched sub or super-class. In order to satisfy the requirements for aggregating these measures, and preserve the [ 0, 1 ] range, the edge count is normalized over the depth of the branch which the two concepts are part of, as per **Definition 22**, giving $CP_{rel} = effort_{norm}$, as shown in Table 14.

**Table 14. Effort with edge count proximity measure**

| Found Relation | Correct Relation | $CP_{pair}$ | $CP_{rel}$ | $CP_{conf}$ | *Comment* |
|---|---|---|---|---|---|
| $LC \equiv TC$ | $LC \equiv TC$ | 1.0 | $effort_{norm}$ | 1.0 | A direct match is best, with an edge count of 0 |
| $LC \equiv TC$ | $LC \Uparrow TC$ | 0.8 | $effort_{norm}$ | 1.0 | Siblings have an edge count of 0, but a lesser proximity than a direct match |
| $LC \equiv TC$ | $LC \sqsubseteq TC$ | 0.4 | $effort_{norm}$ | 1.0 | Matching to a more ambiguous super-class, taking edge count into consideration |
| $LC \equiv TC$ | $LC \sqsupseteq TC$ | 0.6 | $effort_{norm}$ | 1.0 | Matching to a more specific sub-class, taking edge count into consideration |

Finally, the combined proximity measure considers all the information included in the previous measures, as outlined in Table 15 below. When considering how a pair of concepts relate to each other using the $CP_{pair}$ characteristic, a direct match between concepts is rated as best with a value of 1.0, an indirect match between siblings is assigned a value of 0.8, a concept and its more specific sub-class a value of 0.6, and its more general super-class a value of 0.4. When considering the type of relationship the matched concepts $LC_A$ and $LC_B$ have between each other, the effort to go from one to the other is considered, and the $CP_{rel}$ is assigned the value of $effort_{norm}(LC_A, LC_B)$. The particular matching scheme (*ms*) used to create the match is considered by using the associated $ms_{rating}$ value for $CP_{conf}$.

**Table 15. Combined proximity measure**

| Found Relation | Correct Relation | $CP_{pair}$ | $CP_{rel}$ | $CP_{conf}$ | *Comment* |
|---|---|---|---|---|---|
| $LC \equiv TC$ | $LC \equiv TC$ | 1.0 | $effort_{norm}$ | $ms_{rating}$ | A direct match is best; using Matching Scheme rating and a normalized effort |
| $LC \equiv TC$ | $LC \Uparrow TC$ | 0.8 | $effort_{norm}$ | $ms_{rating}$ | Siblings are second best; using Matching Scheme rating and a normalized effort |
| $LC \equiv TC$ | $LC \sqsubseteq TC$ | 0.4 | $effort_{norm}$ | $ms_{rating}$ | Matching to a more ambiguous super-class; using Matching Scheme rating and a normalized effort |
| $LC \equiv TC$ | $LC \sqsupseteq TC$ | 0.6 | $effort_{norm}$ | $ms_{rating}$ | Matching to a more specific sub-class; using Matching Scheme rating and a normalized effort |

## 4.4 Evaluation Use Cases

We now evaluate the test results of our test weather ontology (*ow*) [73] associated with METAR reports [74]. For details regarding the parameters used in these tests, and the computed values discussed in the results section below, see Appendix 1. Once the evaluation measures are analyzed, in section 4.6 we present a real-life use-case that matching the ontologies of two comparable consumer electronics retailers *FutureShop* and *BestBuy*.

**Table 16. Use Case Ontologies**

| Label | Description | Condition Tested |
|---|---|---|
| *wo-wo* | The original weather ontology | 1xx: Match ontology onto itself. |
| *wo-t1* | *wo* with 5 key concepts flattened out, meaning 5 non ground term concepts were removed, and their sub-classes moved up one level. 13 concepts were moved up in total. | 2xx (f): Generalization of direct classification condition of ground terms. |
| *wo-t2* | *wo* with 3 key concepts extended, meaning their sub-classes were grouped under an additional level, affecting 23 direct sub-classes, and creating 6 new concepts. | 2xx (f): Specification of direct classification condition of ground terms. |
| *wo-t3* | *wo* with drastic restructuring, where mid to upper level concepts with several levels of sub-classes are moved, or split up and removed completely. In total, 5 key concepts were moved, 1 concept was removed as all of its sub-classes were moved, affecting 36 direct and indirect sub-classes. 1 new concept was created. | 2xx (c): Loss of consistent classification condition for ground terms, a minor loss of *anchors* with the loss of a concept and the introduction of 1 new concept. Note these concepts were used as classification criteria for creating the decision trees. |
| *wo-t4* | *wo* with 37 key ground terms moved under 5 new super-class concepts. The original 5 super-class concepts were removed. | 2xx (c): Loss of consistent classification condition for ground terms, a major loss of *anchors* with the loss of 5 concepts and the introduction of 5 new concepts. Note these concepts were used as classification criteria used to create decision trees. |
| *wo-t5* | *wo* with several key ground-terms becoming super-classes of other ground-terms. In total 4 ground-terms became sub-classes and 2 concepts were removed. In total 26 concepts were moved. | 2xx (c, f): Loss of consistent classification of 29 ground terms, plus loss of *anchors* with the removal of 2 concepts. |
| *wo-t6* | *wo* with several ground terms becoming super-classes of their siblings. In total, 5 ground terms became super-classes, affecting 29 ground terms. | 2xx (f): Major loss of anchors with the re-classification of ground terms. Some structural consistency exists due to the expansion between siblings. |
| *wo-wo EB* | *wo* ontology, associated with a limited set of data. Instances were selected based on their originating weather station. Station prefixed with "E" for Northern Europe and "B" for Iceland/Greenland and Kosovo. | 1xx: Matching an ontology to itself with lack of records of certain ground terms. This caused many ground terms in *wo* EB not to have sufficient records to generating all decision trees generated in *wo*. |

Before running the METAR data through the decision tree classifier, we must ensure an equal distribution between concepts being classified, and prevent over-fitting for a concept with many records. To achieve this, we create "filler" records for concepts with a low number of records. These "filler" records are copies of existing records already associated with an under represented concept, and are chosen at random. This ensures an equal number and distribution of records representing each concept. Also, we make certain that similar precision rates are used for decision tree leaf nodes by setting the minimum number of records in a leaf node to be 10% of the total number of records associated with the concept being classified. We set the CLASS_NODE_MIN_FACTOR parameter to 0.1 and pass it to the classification portion of the algorithm that creates decision trees. This parameter also controls the maximum number of leaf nodes that will be generated. By setting this parameter to 0.1, we are certain the number of leaf nodes will not exceed 10 or create unnecessarily detailed decision trees. The set of tests in this section use live records from weather stations around the world, often recorded minutes apart for a 24 hour period producing more than 6 million records. Due to test system's CPU and memory limitations, the number of records per ground term has been limited to 30,000. We also select 22 attributes, in combinations of 2 consecutive attributes per decision tree. For example if we had 4 attributes da0, da1, da2 and da3, we would have 5 combinations of attribute pairs, $\{(da_0, da_1), (da_1, da_2), (da_2, da_3), (da_3, da_4), (da_4, da_0)\}$

The use cases for our tests are variations of the original *wo* ontology. Each variation is meant to represent a type of difference that may exist between two heterogeneous ontologies being matched. The tests are described in Table 16, with the *Condition Tested* column indicating which OM test is being adopted, either "Simple Tests (1xx)" or "Systematic Tests (2xx)". We consider the Benchmark as the regular precision and recall, based on proximity measure as described in section 4.3 . Due to the bottom-up nature of the algorithm, heavily based on ground terms, the proximity measures for just the ground terms were noted as "Ground Terms Only". Also, in the algorithm proposed by the thesis, the machine learning algorithm considers siblings as classification criteria for each decision tree in isolation from the rest of the ontology. This ensure that before considering the *data-points* and *rules* of local and target granules, each local and target concept has an equal chance of being matched. For example, $Rain_T$, $Drizzle_T$, $Snow_T$ and $IcePellet_T$ all have an equal chance of being matched to $Rain_L$, as well as to any other local concepts such as $HighWind_L$ or $Tornado_L$. For this reason we also noted "Sibling = Direct" precision and recall, where a sibling match is rated equal to a direct match. By comparing this measure to the "Benchmark" and "Ground Term", we are able to identify what significance siblings have on the matching process.

## 4.5 Evaluation Test Results

We now evaluate the different aspects of the ontology matching algorithm by analyzing the test results in Table 17, and proximity measure analysis with *precision* in Table 21 and *recall* in Table 22 of Appendix 1 on page 86.

**Table 17. Use Case Statistics, with main Precision and Recall Measures.**

| Use Case | Concepts | Derived Concepts | Matched Concepts | Derived Ground Terms | Matched Ground Terms | P | PM | PG | R | RM | RG |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *wo-wo* | 110 | 66 | 60 | 54 | 49 | .91 | .91 | .91 | .55 | .79 | .92 |
| *wo-t1* | 110 | 65 | 46 | 53 | 36 | .71 | .71 | .68 | .42 | .61 | .68 |
| *wo-t2* | 110 | 66 | 54 | 53 | 44 | .82 | .82 | .83 | .49 | .71 | .83 |
| *wo-t3* | 110 | 66 | 54 | 54 | 48 | .82 | .82 | .89 | .49 | .71 | .91 |
| *wo-t4* | 110 | 61 | 33 | 50 | 29 | .54 | .54 | .58 | .30 | .43 | .55 |
| *wo-t5* | 110 | 66 | 40 | 55 | 34 | .61 | .61 | .62 | .36 | .53 | .64 |
| *wo-t6* | 110 | 66 | 52 | 57 | 44 | .79 | .79 | .77 | .47 | .68 | .83 |
| *wo-EB* | 110 | 56 | 14 | 43 | 6 | .25 | .25 | .14 | .13 | .18 | .11 |

**Concepts:** Total number of concepts in the local ontology
**Derived Concepts:** Number of concepts classified using decision trees, and used for matching
**Matched Concepts:** Number of successfully identified matches
**Derived Ground Terms:** Number of ground terms classified using decision trees and used to for matching
**Matched Ground Terms:** Number of ground terms that were successfully matched

**P:** Total Precision
**PM:** Matched Precision
**PG:** Ground Term Precision
**R:** Total Recall
**RM:** Matched Recall
**RG:** Ground Term Recall

Out of the 110 concepts (*Concepts*) in the weather ontology *wo*, only between 56 to 66 (*Derived Concepts*) were classified by both local and target ontologies using decision trees. The intersection of derived concepts is the set of possible matches. This is a limitation of the data, because this algorithm can only match the concepts represented by records in the database. Because there is a great emphasis on ground terms, we also show the number of ground terms that were classified (*Derived Ground Terms*), and the matched (*Matched Ground Terms*). We then show the precision and recall of each. We consider precision and recall only for the concepts that have derived decision trees. These measures would increase if the data were available to classify all concepts.

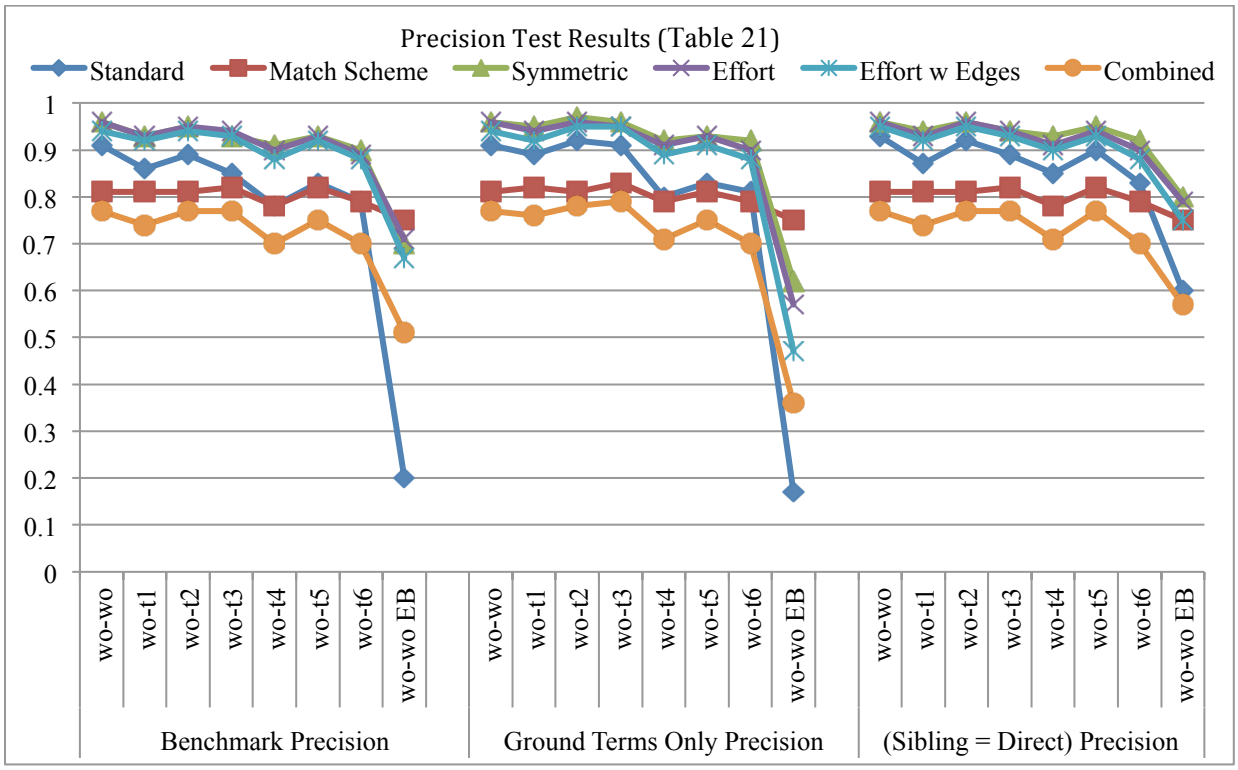



**Figure 24. Test results of *precision* vs. use cases, for each proximity measure.**

Figure 24 above, illustrates the precision of each use case, for each proximity measure, which is the rate of true positives found. It should be noted that none of the tests achieved 100% precision. It may be expected that at least, the *wo-wo* test would return 100% precision, or recall in Figure 25 below. The reason for this is the level of generality which was introduced to the granule rules and data-points discussed in sections 3.7 and 3.8. This generality blurred the discrete lines between concepts, for the sake of successful matches between two heterogeneous ontologies.

The best precision for proximity measures was achieved by the "Effort", "Effort with Edges" and "Symmetric" measures. This indicates that the structural relationships that these measures are based on played a highly significant role in finding correct matches. This is especially noticeable when compared to the "Standard" measure, which only considered direct matches, and has a lower precision than the other three. They were also affected negatively by the loss of anchors and ground terms in the *wo-EB* use case, especially in the "Ground Term" rating. Each of the three measures also produced close results, with parallel variation for each use case. Due to the similarity of *precision* for each indirect match measure, mainly "Effort", "Effort with Edges" and "Symmetric", no significant benefit was achieved by considering whether a generalized super-class or specialized sub-class was found, nor the effort distance between them. This was due to the small number of indirect matches found

with high effort scores. Still, there was enough indirect matches for these three measures to be higher than the ”Standard” measure. Of the three, there was no major difference between the “Benchmark”, and “Ground Term" rating, while the “Sibling = Direct” was slightly better on the *wo-t4*, *wo-t5*, and *wo-t6* use cases. The “Effort with Edges” measure also suffered the largest drop for the *wo-EB* use case, especially for the “Ground Term” rating. Again, this is because the “Ground Term” rating is directly affected by the number of ground terms and their matches, that were lost in *wo-EB*.

The next best precision was achieved by the “Standard” measure. This measure considered only direct matches, without taking into consideration any other proximity measures. When compared to the indirect measures described above, mainly “Effort”, “Effort with Edges”, and “Symmetric”, we can see the improvement the addition of indirect matches had. While starting off high at above 0.91 for a match with the ontology itself (*ow-ow*), we see the measure drop for the “Benchmark” rating. It should be noted that the *wo-t1* use case had the least number of changes, and resembled the original ontology most. The “Ground Term” rating has a steady precision until use case *wo-t4*, when it drops to around 0.81 through to use case *wo-t6*. As can be observed, the three indirect measures held up much better to the differences in the ontologies, than direct measure ”Standard.” The *wo-t3* and *wo-t6* use cases, which saw a change in ground terms being shifted within the ontology, have an advantage for the “Sibling = Direct” tests. This indicates a tolerance for specialized ground terms created by the expansions in *wo-t3* and *wo-t6*. Finally, we see that the “Standard” measure experienced the biggest drop for *wo-EB* use case. The loss of data records to build decision trees greatly impacted this measure. The impact was not as great when considering siblings as direct matches, as per the precision of “Sibling = Direct” tests. While many anchors were lost in the *wo-EB* ontology, the ones that remained were structurally similar to the *wo* ontology, and remained at a high precision for the available concepts.

The next best measure is “Match Scheme”, which considers the matching scheme ratings only. This measure is not affected by the types of matches, and considers all direct and indirect matches equally. It does highlight which matching schemes are being utilized. For example, the steady rating with an average of 0.8 indicates a high number of “Gap Match” matches, which have a rating of 1.0. This match scheme is offset by lower rated schemes, lowering it to 0.8. By evaluating the matches themselves, we found around 1.5 direct “Gap Parent” matches for every 1 “Close Match”, with a rating of 0.5. By averaging the combinations of these two, the score was calculated as 0.8. The final measure is the “Combined” proximity measure, which considers all characteristics of MCP. Its low precision can be attributed to being the product of all other proximity characteristics. When compared

to the other measures, it does parallel the other measures as a whole. As a result, we conclude that it can in fact be used as a suitable measure that encompasses the other characteristics. It would off course need to be compared to other "Combined" measures, and not the other specialized measures as presented here.

Finally, we mentioned again the observed drop in the *wo-EB* use case for all measures, which was built with records of only a few ground terms. Based on the test results, it seems that the "Sibling = Direct" result was least sensitive to this lack of data, still managing to keep a precision above 0.4 for all measures, due to the differences being more specialized ground terms. By looking at the *recall* rate next, we see how this is proportional to the low recall for this use case.

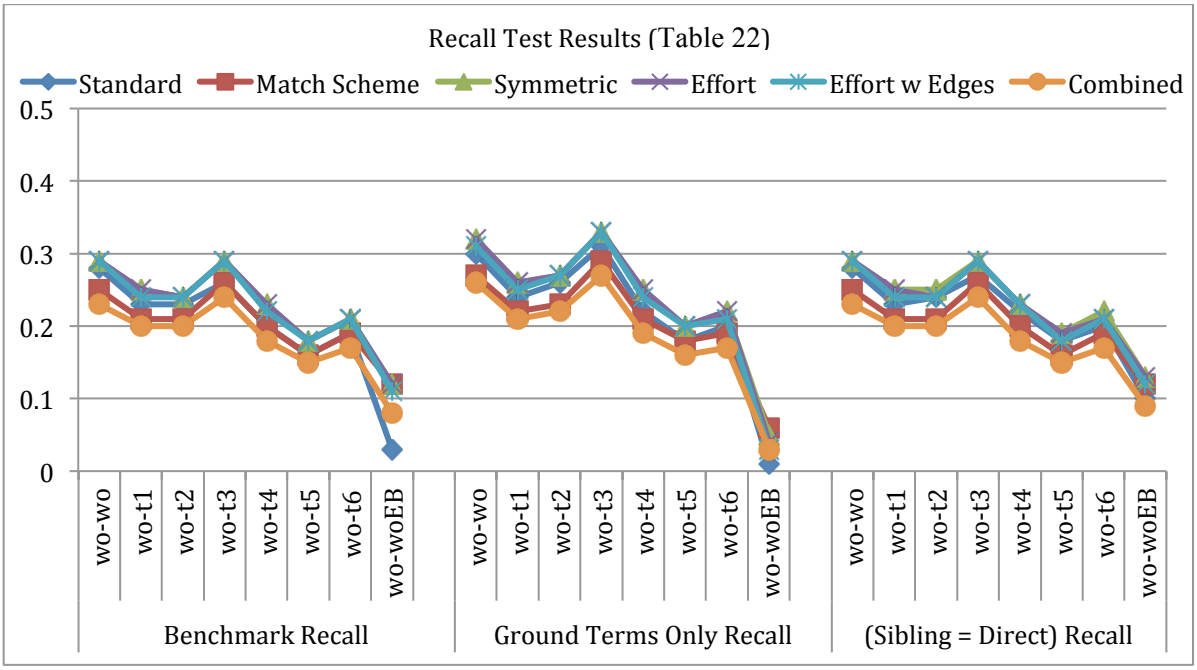



**Figure 25. Test results of *recall* vs. use cases, for each proximity measure.**

In Figure 25, the *recall* rate identifies the percentage of returned matches. The low recall rate, mostly below 0.4 indicates the user is given several choices to choose from, but these choices have the correct match most of the time, as indicated by the high *precision* scores above. This low score is most likely due to the low number of ground terms making up the entire match space. This hypothesis is supported by two observations. Firstly, the "Ground Terms" tests had a noticeably better overall recall rate, if by a small margin. The reason for this is that of the recalled matches, the majority of them were ground terms, as indicated by the "Matched Ground Terms" column in Table 17. This is because of the consistency in the data, and the direct association of data with ground terms. This again

supports the bottom up *anchor* creation approach favored by this matching algorithm. Having different ontologies, the positive affect that consistency in the data has on concept matching is apparent in this result. Secondly, consider the drastic drop of recall rate for the *wo-EB* use case for all three tests, "Benchmark," "Ground Term," and "Sibling = Direct". *wo-EB* had significantly less ground terms associated with data records. This negatively affected all three tests, especially "Ground Terms", having the lowest.

Similarly to the *precision* results, the "Effort", "Effort with Edges", and "Symmetric" measures had the highest *recall* rate, reiterating the conclusion that these measures had a significantly positive affect on the retrieval of correct matches. The "Match Scheme" measure again has the next best results, followed closely by "Combination" measure. Similarly to the *precision* rate patterns, each measure was in parallel with the other, showing a consistency in the proximity measures used. This observation indicates correct proximity measures were chosen, which are affected similarly by the matching results.

## 4.6 Real-Life Use Case And Test Results

In this section we present a real-life use case where we match the ontologies of two consumer electronics retailers *FutureShop*[44] and *BestBuy*[45]. The results are presented in a similar format to the weather ontology in section 4.5, with the exception that results and evolution measure are presented for one test, matching *FutureShop* concepts to *BestBuy* concepts. In contrast, section 4.5 presented results for six tests, comparing the weather ontology to 7 variations of the original ontology. The purpose of the previous tests was to introduce the evaluation measures, and demonstrate their performance when matching ontologies with various types of differences between them. The purpose of this section is to apply the ontology matching algorithm and evaluate the algorithm on a real-life use-case.

The two ontologies and associated data records are both taxonomies extracted from their respective websites. Only the *Computer* categories were used for the test, as it contained the most number of products. The taxonomy was built from each retailer's hierarchical catalog menu structure, with the ground terms being the last menu item. These menu items were associated with a list of products. Those products were then extracted from the site along with their associated attributes accessible on

[44] http://www.futureshop.ca/en-ca/home.aspx
[45] http://www.bestbuy.ca/Search/SearchResults.aspx?

the product detail page. Because the two retailers are owned by the same company, they share the same named attributes, although the taxonomies and the majority of products are different. Evry combination of the 95 selected attributes was used to generate decision trees, although any trees with a depth of 0, a single root node, were excluded from the matching process.

The *FutureShop* taxonomy, and specifically the *Computer* subset has a shallow depth of 3 levels and contains 89 concepts, 73 of which are ground-terms. In contrast the *Computer* subset of the *BestBuy* taxonomy has a depth of 4 levels, with 232 concepts 185 of which are ground-terms.

**Table 18. FutureShop vs. *BestBuy* Precision and Recall Measures summary.**

| Use Case | Concepts | Derived Concepts | Matched Concepts | Derived Ground Terms | Matched Ground Terms | P | PM | PG | R | RM | RG |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *fs-bb* | 89 | 37 | 24 | 30 | 16 | .64 | .64 | .53 | .41 | .41 | .34 |

In Table 18, we present the summary of the test results. Here we see that out of the 89 *FutureShop* concepts, 37 had enough data records to be associated with decision trees. 30 of the *FutureShop* concepts associated with decision trees were ground-terms. In total, 24 *FutureShop* concepts were successfully matched with their *BestBuy* counterpart, 16 of which were ground-terms. The matching algorithm performed relatively well, reaching 64% precision with a 41% recall rate.

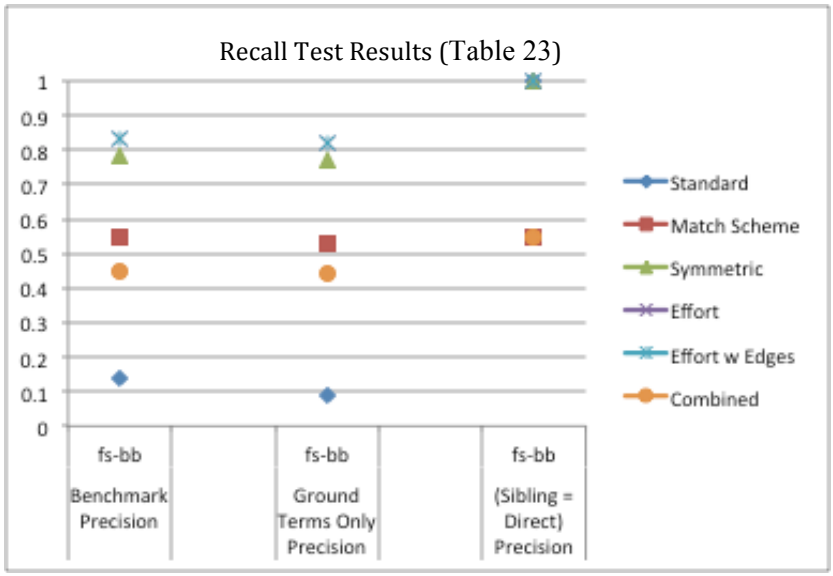



**Figure 26. Test results of *precision* for *FutureShop* vs. *BestBuy*, for each rating (x-axis) and proximity measure (legend on right).**

Figure 26 above illustrates the *precision* achieved by each proximity measure when matching the *FutureShop* concepts to the *BestBuy* taxonomy. The *FutureShop* taxonomy is relatively shallow being only 3 levels deep, which affects the measures in the following way. The "Effort" and "Effort with Edges" measures both achieved equal precision scores of 0.83 for the "Benchmark" rating, followed closely by "Symmetric" with 0.78. We can also see a score of 1.0 for the "Sibling = Direct" rating for "Effort", "Effort with Edges" and "Symmetric" measures. Because of the low depth, all of the indirect matches were made to siblings of the correct match. Also, because the matched siblings are at the same level as the correct match, the "Effort" and "Effort with "Edges" gave the same score. The number of edges between sibling classes is zero, making the "Effort" and "Effort with Edges" indistinguishable for the "Sibling = Direct" rating.

Similarly to the Weather Ontology test results in section 4.5, the "Effort", "Effort with Edges" and "Symmetric" measures have the highest scores, reiterating again the positive affect incorporating structural relationships has on the matching process. The second highest measure is the "Matching Scheme" measure with a score between 0.53 and 0.55 for the "Benchmark", "Ground-Terms" and "Sibling = Direct" ratings. The majority of the matches were created using the "Close Match" matching scheme with a $ms_{rating} = 0.5$, which averaged out to between 0.53 and 0.55. This scheme considers high *MRank* values between concepts before making a match, as per **Definition 15** on page 56. Due to the low number of data records extracted for some concepts (between 10 and 100), the minimum *MRank* threshold was set low to 0.2. This essentially loosened the criteria for considering a match correct by the "Close Match" scheme. In contrast, the Weather Ontology tests used live records from weather stations often resulting in up to 30,000 records per ground-term. As a result the weather tests had a much higher "Close Match" threshold of 0.9.

The "Combined" proximity measure averaged between 0.44 and 0.55 for each rating. As mentioned in the previous section, this measure is essentially a combined product of the "Match Scheme" and "Effort with Edges" measures, and is meant to measure the algorithms performance using a single value, incorporating the other proximity measures into one. As expected this measure is lower then the other ones because it is a product of decimal numbers in the [0, 1] range, as per Equation 10 on page 62, considering several values listed in Table 15 on page 67. What is important is whether they are parallel to the other measures. As in the weather tests, this measure is parallel to the other scores, meaning it successfully reflects the other measures.

Finally the "Standard" proximity measure is the lowest at 0.14 and 0.09 for the "Benchmark" and "Ground-Term" ratings. This measure only considers direct matches as correct, and all non-direct matches as incorrect. This result indicates that in general there was a low number of direct matches. Also, due to the slightly higher score for the "Benchmark" rating, more non-ground-term concepts were matched using this measure then ground-terms, essentially indicating that more non-ground-term *FutureShop* concepts were matched to their direct *BestBuy* counterpart then ground-terms. Also, the "Standard" measure for the "Sibling = Direct" rating is equal to 1.0, echoing the "Effort", "Effort with Edges" and "Symmetric" proximity measures. This again indicates that despite the low number of direct matches, the algorithm found the correct concept's sibling instead.

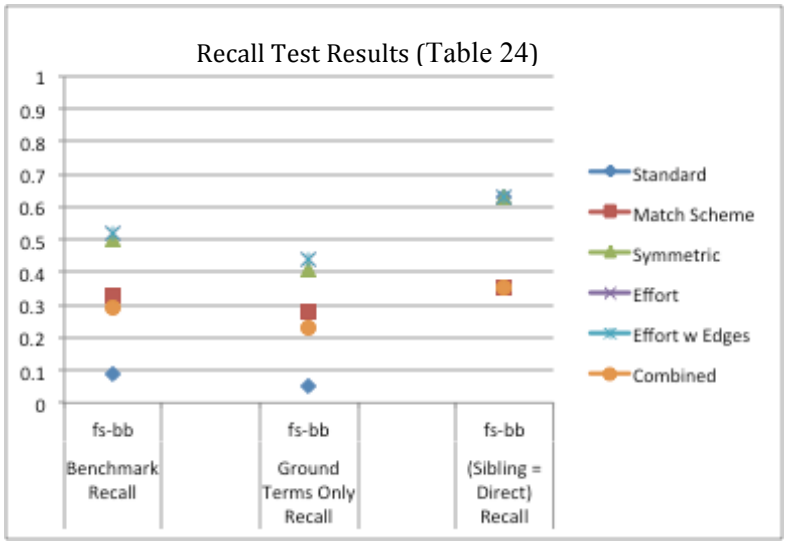



**Figure 27. Test results of *recall* for *FutureShop* vs. *BestBuy*, for each rating (x-axis) and proximity measure (legend on right).**

In Figure 27, the *recall* rate represents the number of returned results as a percentage of all possible correct matches. For the "Benchmark" rating, the highest score is achieved by "Effort" and "Effort with Edges" with 0.49, followed closely by the "Symmetric" measure with 0.46. Similarly to the *precision* results above, the highest scores were achieved by these three. The "Matching Scheme" proximity measures again achieved the second highest score with 0.33. The "Standard" measure was the lowest again indicating that few direct matches were found. The combination of a high precision core with a low recall score indicates that although the direct match was not the top match, it did exist in the suggested match results. This pattern was repeated for the "Ground-Term" and "Sibling = Direct" ratings, with the exception for the "Standard" measure. As with the *precision* values, the

"Standard" measure equals that of "Effort", "Effort with Edges" and "Symmetric". This again indicates that majority of indirect matches were made with a sibling of the correct match.

## 4.7 Ontology Considerations

As can be deduced from the test results, the ontologies being matched by this algorithm must be consistent in some regards, and can differ in others. This section describes some considerations and restrictions put on potentially matching ontologies. For example, consistency in the data is an important factor, especially when there are differences in the structure of the ontology.

Ontology design is very closely tied to a user's interpretation of their domain [8]. For this reason, equivalent concepts can be structured differently in similar domains. While the literature [8] describes different considerations, we concentrate on the aspects directly affecting the usability of this algorithm.

Recall the local ontology $O_L$ and target ontology $O_T$, illustrated again in Figure 28 (a) and (b).

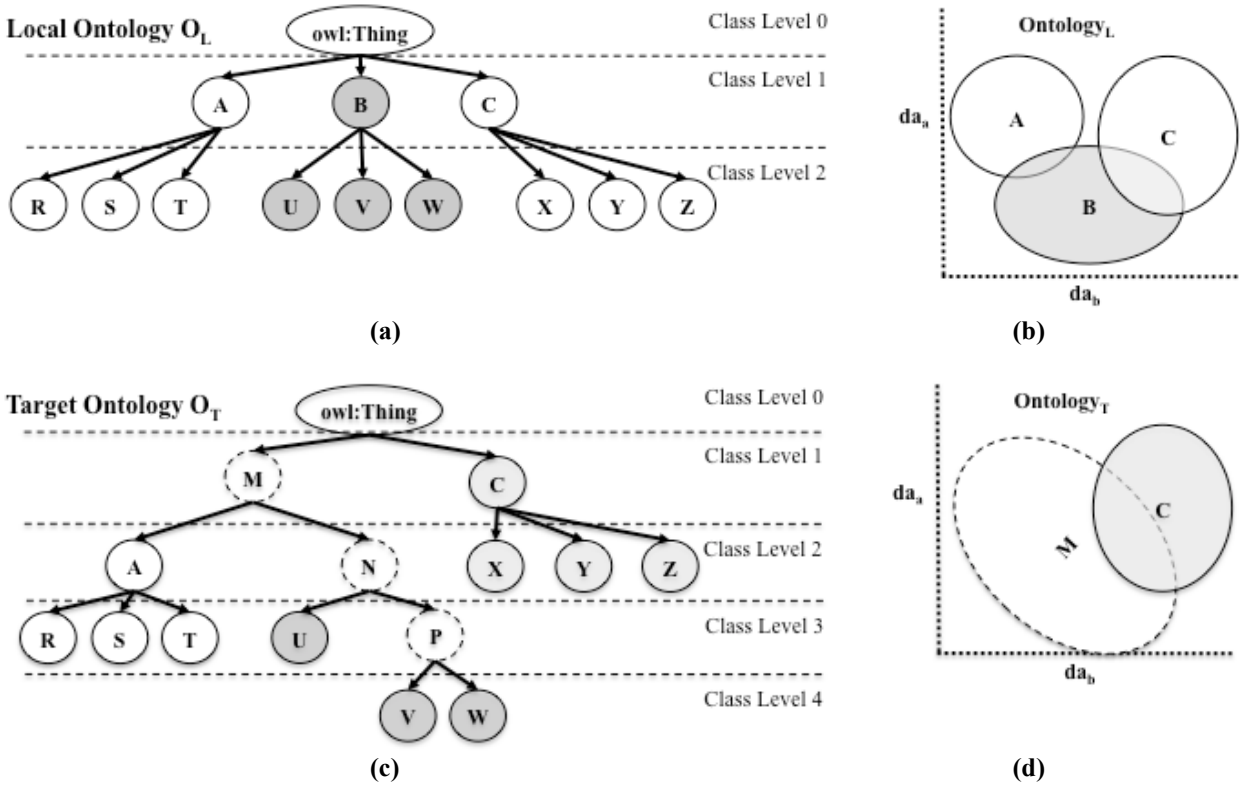



**Figure 28. $O_L$ (a) and $O_T$ (c) ontologies, with corresponding top level data clusters, (b) and (d) .**

Recall also from section 3.5, that the similarities between the $O_L$ and $O_T$ ontology, mainly that the sub-classes of $A$ ($A_L$ and $A_T$) and $C$ ($C_L$ and $C_T$) are similarly classified, and that $B_L$ is similar to the combination of $N_T$ and $P_T$. The decision tree algorithm works by taking a particular set of records, the entire sample initially, and determines which property has the highest information gain to classify each class. When choosing an attribute automatically based on its contribution to classification, various rankings can be used. The data-mining tool WEKA [25], provides several algorithms, such as information gain, entropy, and principal component. The J48[46] decision tree algorithm uses the information gain[47] measure to calculate property relevance. The tree is then split on that property, generating two branches ( $\leq$ and $>$ ) for numeric attributes, and multiple branches for nominal attributes. The key factor to observe here is that the decision tree is directly impacted by the set of classes being classified, and the records themselves, neither of which can be guaranteed to be exactly the same in individual systems. These two factors impact what property is chosen for each sample of records, at different stages of the decision tree creation. This in turn has an impact on several aspects of the ontology matching process.

Firstly, the local and target databases, $\mathcal{DB}_L$ and $\mathcal{DB}_T$ respectively, must illustrate some level of consistency, in other words, they must be representing similar observations. Secondly, the ontologies, and sub-class relations, must also show some level of consistency. Specifically, the direct sub-classes of a concept must be similar, in order to build similar clusters on a 2-dimensional plane. To prevent over-fitting by any individual sub-class, the distribution of records for each sub-class must be similar [25]. Take for example the Linear Ontology $O_{Linear}$, illustrated in Figure 29 (a) and (b).

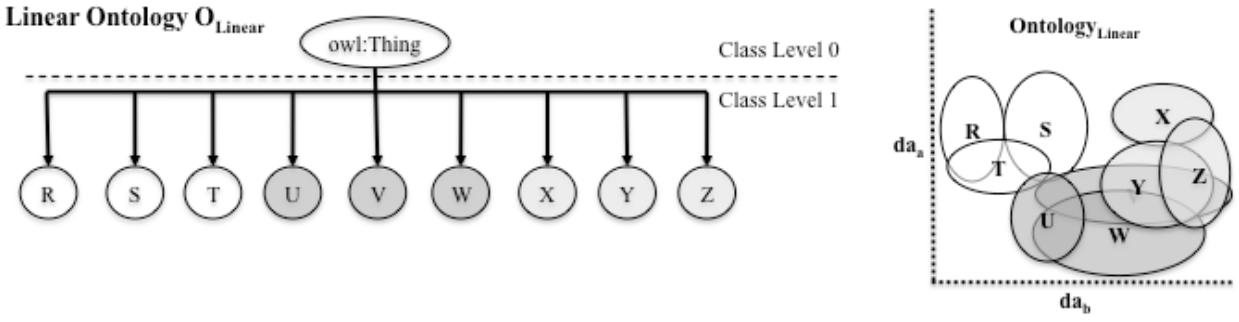



**Figure 29. Linear Ontology Example ($O_{Linear}$) (a) and a 2-dimensional clusters (b).**

[46] The WEKA 3.6.0 module *weka.classifiers.trees.J48* used to implement the C4.5 classification algorithm.
[47] Definition taken from the WEKA 3.6.0 module *weka.attributeSelection.InfoGainAttributeEval*

When viewing the cluster representation in Figure 29 (b), we can see the individual ground terms are clustered in a similar manner to $O_L$ and $O_T$. Despite this, the decision tree generated for it will look differently, because there are more sub-classes to classify at once, {*R, S, T, U, V, W, X, Y, Z*} as appose to grouping them under their super-classes *A*, *B*, or *C* in $O_L$, and *A*, *C*, *N*, *M*, or *P* in $O_T$. The resulting rules for each class, and associated granules may in fact be overlapping, but because more classes are involved, some boundaries will be larger or smaller. As a result, the individual branches, representing a decision tree leaf node will contain different rules. These new rules may or may not overlap with another ontology's rules. Figure 30 below illustrates the rules a decision tree may build to represent $LC_C$, $TC_C$, and *owl:Thing*. Some of the shaded rules do overlap, but others do not.

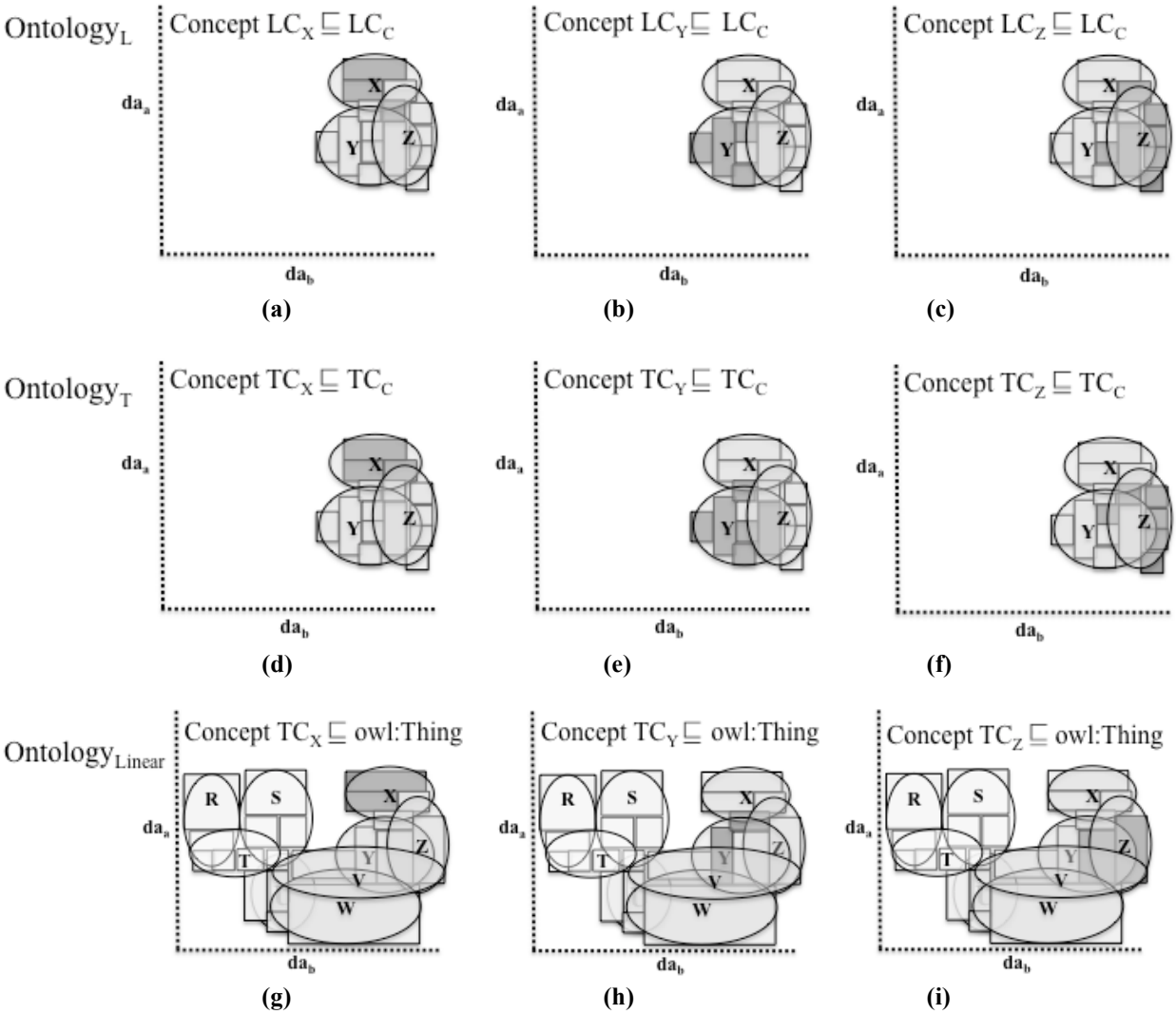
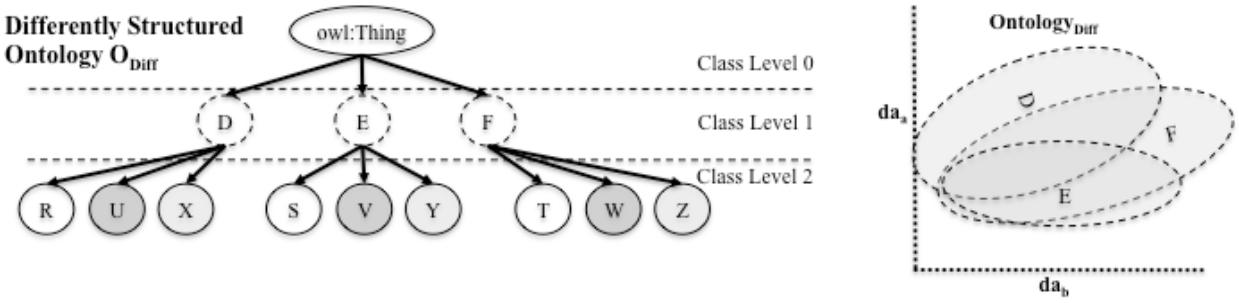



**Figure 30. $LC_A$, $TC_A$ , and *owl:Thing* granule matches: (a) $\doteq$ (d) $\doteq$ (g), (b) $\doteq$(e) $\doteq$(h), and (c) $\doteq$ (f) $\doteq$ (i).**

Another example of an ontology that differs in structure is one that classifies sub-classes differently. Figure 31 below illustrates such an ontology, $O_{Diff}$. Notice that $O_{Diff}$ looks similar to $O_L$, but the ground terms are actually classified under different super-classes. In Figure 31 (b), we see how at level 1, neither $D_{Diff}$, $E_{Diff}$ nor $F_{Diff}$ look similar to $LC_A$, $LC_B$, or $LC_C$ in Figure 28 (b).




**(a)** **(b)**

**Figure 31. Differently Structured Ontology Example ($O_{Diff}$) (a) and a 2-dimensional cluster (b).**

To see similarities, we must look at the ground terms, as illustrated in Figure 32 below. When using the same attributes $da_a$ and $da_b$, the clusters look very different for each super-class {$D_{Diff}$, $E_{Diff}$ , $F_{Diff}$ } and {$LC_A$, $LC_B$, $LC_C$}. With clear boundaries and different overlapping areas between clusters, different rules will be generated in the decision trees for each of these parent nodes. For this reason, instead of using all attributes at once, multiple 2-dimensional trees are built, using a combination of every property available in the database. There is a better chance of identifying overlapping areas between different combinations of attributes, for different datasets.

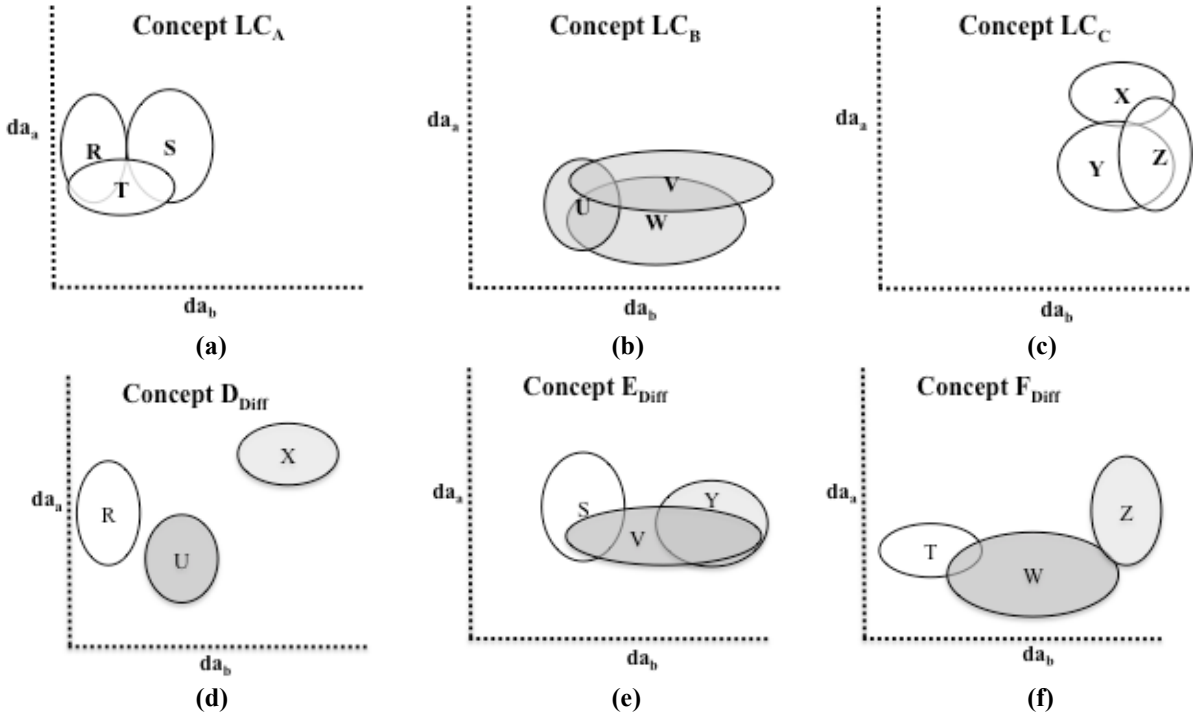



**Figure 32. 2-dimensional, Level 2 sub-class clusters for $O_L$ and $O_{Diff}$.**

Finally, the decision tree algorithm incorporates a level of randomness to its decision making. This allows it to process missing values for attributes in the records it uses for classification [25], but at the same time, skewing classification results. As mentioned in section 2.2, the *bagging* technique, which stands for bootstrap aggregation, is used to produce multiple decision trees for each set of data. The *bagging with cost* technique is then used to combine all trees produced by *bagging*, and create a stabilized tree, which encompasses all information of the original trees. These extra steps reduce the randomness affect of noisy attributes with missing values. Section 2.2 on page 9 discusses machine learning and the motivation for choosing these techniques, in greater detail.

# Chapter 5. Conclusion

## 5.1 Contribution

This thesis proposes a novel approach to ontology matching, mainly one which is solely based on concrete data that represents ontological concepts. We demonstrate how the algorithm applies machine learning techniques to represent semantics between concepts, and inductively build decision trees using those semantics. The decision trees are then represented in OWL 2 syntax as rules. The Description Logic reasoner HermiT is utilized to perform the matching process. We then applied a novel match ranking algorithm to select the best matches, and extend existing ontology matching measures to evaluate the results.

We present the algorithm and demonstrate several measures that can be used to evaluate it, with a set of use cases made up of specifically altered *weather* target ontologies, and match real-life ontologies from the BestBuy and FutureShop retailers. These use cases demonstrate the benefits and draw backs of this algorithm. The algorithm avoids differences in labels, which can vary significantly between ontologies, and places a lower priority on structural differences. We achieve this by concentrating on the data representing the concepts being matched. By utilizing machine learning, we inductively build decision trees that identify the conditions that differentiate a set of concepts from their ontological siblings. Note that siblings rely on a highly localized structural similarity, as oppose to having to consider the entire ontology as a whole. Using these conditions, we build a set of granules, with associated rules and data-points. These are then represented in OWL 2 syntax, to make them available to a wider audience. By using the algorithm presented here, the local system as well as a target system can now match their ontologies by sharing these granules, and identify overlapping rules and data-points between their local and target ontologies

Another key benefit of the algorithm is that it takes advantage of the consistency in data to introduce consistency between two separate ontologies, which may differ greatly. However one main drawback of this association with data is that the only concepts that can be matched are ones that are represented by the data. For example, if the dataset being used does not have records that represent a particular concept in the ontology, rules for that concept will not be generated, and it will not be matched. However, the assumption is that the most important concepts to match are ones which do exists in the dataset, and are used by the local system. This however is implementation dependent, and will reduce the applicability of this algorithm for implementations affected by this drawback.

The evaluation and proximity measures that are proposed and tested by this thesis establish a method for analyzing the matching preformed by our algorithm. By considering various aspects of the algorithm, and identifying a set of generalized proximity measures to analyze each aspect, we show the types of differences which impact the successful matching of concepts.

## 5.2 Future Work

The approach to ontology matching in this thesis is a novel approach to incorporating data and machine learning algorithms, and has many exciting directions and possibilities for enhancements. The matching algorithms in the OM Workshop [72] competition all use a combination of lexical and structural techniques, and run through several iterations refining their matches by incorporating various new techniques. This algorithm only deals with creating the initial *anchors*. Implementing more complex refinement iterations would improve the overall match rate, by incorporating existing matching techniques. Also, extending the expressivity of ontologies to include new relationships beyond super/sub-class would be an ideal extension to this algorithm. Finding correspondences using more complex semantics would increase the algorithm's applicability against real-life use cases. More sophisticated logical evaluation and verification algorithms would improve the precision and recall of the algorithm. For example, a check for contradictory super/sub-classes would ensure structural characteristics of concepts are preserved, and the generality of super-classes does not overwhelm the specialization of sub-classes.

The METAR format used in the weather use case demonstrated that a single database record, with multiple attributes, could represent multiple ontology concepts. Composite attributes, which form a new property when used together, require a more abstract approach. By incorporating not just concept hierarchical analysis, but also property hierarchies, more complex database designs could be used to associate ontological concepts with data records. In the solution presented here we resulted to normalizing the database. However for larger databases, with multiple tables and a greater number of attributes this could cause a scaling problem, due to the duplication of values during normalization. An enhanced hierarchical property association would allow the algorithm to scale up to more complex database designs.

While the presented extensions of the matching algorithm are novel, the generated OWL 2 axioms themselves can be applied to other research area. For example, executing queries over the semantic web [5] requires not just knowledge of the external system's schema, but also its ontology. By

creating a semantically enhanced query, generated from the semantically enhanced decision trees, it may be possible to apply the rules-portion of the generated granules to query a target system using the distributed target rules. The query itself can be reconfigured to abide by the target database system's schema, through query rewriting.

# Appendix 1. Test Parameters and Evaluation Data

## Parameters

For the use cases in Chapter 4, we use the following parameter values needed by the algorithm. A detailed description of the parameters is found in the glossary under "Acronyms and Variable Name Conventions". Any parameters used in pairs are described together.

**Table 19. Parameters used by the matching algorithm.**

| | Value | | | |
|---|---|---|---|---|
| **Field** | ***wo-**** | ***fs-bb*** | **Purpose** | **Comment** |
| *CLASS_NODE_MIN_FACTOR* | 0.1 | 0.1 | A minimum number of records in a decision tree leaf node ensures equal precision of each leaf node. | 0.1 was chosen to have at most 10 leaf nodes. For more complex decision tree, this parameter can be reduced. |
| *RULE_CLUSTER_BUFFER* | 0.1 | 0.1 | To generalize the rules built by the decision tree algorithm, this buffer is used to expand granule rules and decrease data-points. | For domains which require much stricter guidelines and less flexibility, a lower value is beneficial. |
| *GAP_CHECK_MIN* | 0.7 | 0.9 | Minimum *MRank* of a match before it is check for a "Gap Match" *GM*. | Low values ensure only close gaps were considered between high ranking matches [*MRank* ≥ 0.5 (0.7 - 0.2)]. For example match(0.6, Cloud) and match(0.4, *Heat-Index*) would not be considered for the *GM* check, as the higher of the two match(0.6, *Cloud*) had too low of a ranking (0.6) to be considered. |
| *GAP_MIN* | 0.2 | 0.7 | The gap used to identify a "Gap Match" *GM*. | |
| *TOP_RANK_MIN* | 0.9 | 0.2 | Used to identify a "Close Match" *CM*. | A higher value indicates a stricter criteria for a match. |
| *TOP_P2P_RANK_MIN* | 0.6 | 0.6 | The *MRank* minimum threshold to identify a parent-to-parent match after a successful *GM* was found between their sub-classes. | A more flexible criteria would require a smaller value. |

The following tables contain the results of the matching algorithm tests discussed in Chapter 4.

Table 20 contains the Weather Ontology test Concept counts and main *Precision* and *Recall* for each category, as described below here, and elaborated on in Chapter 4. Table 23 and Table 24 on page 90 contain the *Precision* and *Recall* for the *FutureShop* vs. *BestBuy* test.

**Concepts:** Total number of concepts in the local ontology

**Derived Concepts:** Number of concepts classified using decision trees, and used for matching

**Matched Concepts:** Number of successfully identified matches

**Derived Ground Terms:** Number of ground terms classified using decision trees and used to for matching

**Matched Ground Terms**: Number of ground terms that were successfully matched

**P**: Total Precision

**PM:** Matched Precision

**PG**: Ground Term Precision

**R**: Total Recall

**RM**: Matched Recall

**RG**: Ground Term Recall

**Table 20. Use Case Statistics, with main Precision and Recall Measures.**

| Use Case | Concepts | Derived Concepts | Matched Concepts | Derived Ground Terms | Matched Ground Terms | P | PM | PG | R | RM | RG |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *wo-wo* | 110 | 66 | 60 | 54 | 49 | .91 | .91 | .91 | .55 | .79 | .92 |
| *wo-t1* | 110 | 65 | 46 | 53 | 36 | .71 | .71 | .68 | .42 | .61 | .68 |
| *wo-t2* | 110 | 66 | 54 | 53 | 44 | .82 | .82 | .83 | .49 | .71 | .83 |
| *wo-t3* | 110 | 66 | 54 | 54 | 48 | .82 | .82 | .89 | .49 | .71 | .91 |
| *wo-t4* | 110 | 61 | 33 | 50 | 29 | .54 | .54 | .58 | .30 | .43 | .55 |
| *wo-t5* | 110 | 66 | 40 | 55 | 34 | .61 | .61 | .62 | .36 | .53 | .64 |
| *wo-t6* | 110 | 66 | 52 | 57 | 44 | .79 | .79 | .77 | .47 | .68 | .83 |
| *wo-EB* | 110 | 56 | 14 | 43 | 6 | .25 | .25 | .14 | .13 | .18 | .11 |

The following two tables contain detailed results for each proximity measure in the Weather Ontology tests discussed in Chapter 4.

**Table 21. Precision Evaluation Measures.**

| Benchmark | Use Case | Standard | Match Scheme | Symmetric Effort | | Effort w Edges | Combined | Use Case Average |
|---|---|---|---|---|---|---|---|---|
| | *wo-wo* | 0.83 | 0.73 | 0.87 | 0.87 | 0.86 | 0.7 | **0.81** |
| | *wo-t1* | 0.61 | 0.57 | 0.66 | 0.66 | 0.65 | 0.52 | **0.61** |
| | *wo-t2* | 0.72 | 0.67 | 0.78 | 0.78 | 0.77 | 0.63 | **0.73** |
| | *wo-t3* | 0.7 | 0.67 | 0.76 | 0.77 | 0.76 | 0.63 | **0.72** |
| | *wo-t4* | 0.42 | 0.42 | 0.49 | 0.49 | 0.48 | 0.38 | **0.45** |
| | *wo-t5* | 0.5 | 0.49 | 0.57 | 0.56 | 0.56 | 0.46 | **0.52** |
| | *wo-t6* | 0.62 | 0.63 | 0.71 | 0.7 | 0.69 | 0.55 | **0.65** |
| | *wo-wo EB* | 0.05 | 0.19 | 0.17 | 0.18 | 0.17 | 0.13 | **0.15** |
| **Benchmark Averages** | | **0.56** | **0.55** | **0.63** | **0.63** | **0.62** | **0.50** | **0.58** |

| Ground Terms Only | Use Case | Standard | Match Scheme | Symmetric Effort | | Effort w Edges | Combined | Use Case Average |
|---|---|---|---|---|---|---|---|---|
| | *wo-wo* | 0.83 | 0.74 | 0.87 | 0.87 | 0.86 | 0.7 | **0.81** |
| | *wo-t1* | 0.6 | 0.55 | 0.64 | 0.64 | 0.63 | 0.52 | **0.60** |
| | *wo-t2* | 0.76 | 0.67 | 0.8 | 0.8 | 0.79 | 0.65 | **0.75** |
| | *wo-t3* | 0.81 | 0.74 | 0.85 | 0.85 | 0.84 | 0.7 | **0.80** |
| | *wo-t4* | 0.47 | 0.46 | 0.53 | 0.53 | 0.52 | 0.41 | **0.49** |
| | *wo-t5* | 0.51 | 0.5 | 0.58 | 0.57 | 0.56 | 0.46 | **0.53** |
| | *wo-t6* | 0.63 | 0.61 | 0.71 | 0.7 | 0.68 | 0.54 | **0.65** |
| | *wo-wo EB* | 0.02 | 0.1 | 0.09 | 0.08 | 0.07 | 0.05 | **0.07** |
| **Ground Term Averages** | | **0.58** | **0.55** | **0.63** | **0.63** | **0.62** | **0.50** | **0.59** |

| (Sibling = Direct) | Use Case | Standard | Match Scheme | Symmetric Effort | | Effort w Edges | Combined | Use Case Average |
|---|---|---|---|---|---|---|---|---|
| | *wo-wo* | 0.84 | 0.73 | 0.88 | 0.87 | 0.86 | 0.7 | **0.81** |
| | *wo-t1* | 0.62 | 0.57 | 0.66 | 0.66 | 0.65 | 0.53 | **0.62** |
| | *wo-t2* | 0.76 | 0.67 | 0.79 | 0.78 | 0.77 | 0.63 | **0.73** |
| | *wo-t3* | 0.73 | 0.67 | 0.77 | 0.77 | 0.76 | 0.63 | **0.72** |
| | *wo-t4* | 0.46 | 0.42 | 0.5 | 0.49 | 0.48 | 0.38 | **0.46** |
| | *wo-t5* | 0.55 | 0.49 | 0.58 | 0.57 | 0.56 | 0.46 | **0.54** |
| | *wo-t6* | 0.65 | 0.63 | 0.72 | 0.71 | 0.7 | 0.56 | **0.66** |
| | *wo-wo EB* | 0.15 | 0.19 | 0.2 | 0.2 | 0.19 | 0.14 | **0.18** |
| **(Sibling=Direct) Averages** | | **0.60** | **0.55** | **0.64** | **0.63** | **0.62** | **0.50** | **0.59** |

| **Precision Average** | **0.58** | **0.55** | **0.63** | **0.63** | **0.62** | **0.50** | **0.58** |
|---|---|---|---|---|---|---|---|

**Table 22. Recall Evaluation Measures.**

| Benchmark | Use Case | Standard | Match Scheme | Symmetric Effort | | Effort w Edges | Combined | Use Case Average |
|---|---|---|---|---|---|---|---|---|
| | *wo-wo* | 0.72 | 0.64 | 0.76 | 0.75 | 0.75 | 0.6 | **0.70** |
| | *wo-t1* | 0.52 | 0.49 | 0.57 | 0.56 | 0.55 | 0.45 | **0.52** |
| | *wo-t2* | 0.63 | 0.58 | 0.68 | 0.67 | 0.67 | 0.54 | **0.63** |
| | *wo-t3* | 0.61 | 0.58 | 0.66 | 0.67 | 0.66 | 0.54 | **0.62** |
| | *wo-t4* | 0.34 | 0.34 | 0.39 | 0.39 | 0.38 | 0.3 | **0.36** |
| | *wo-t5* | 0.44 | 0.43 | 0.49 | 0.49 | 0.48 | 0.4 | **0.46** |
| | *wo-t6* | 0.54 | 0.54 | 0.62 | 0.61 | 0.6 | 0.48 | **0.57** |
| | *wo-wo EB* | 0.04 | 0.14 | 0.13 | 0.13 | 0.12 | 0.09 | **0.11** |
| **Benchmark Averages** | | **0.48** | **0.47** | **0.54** | **0.53** | **0.53** | **0.43** | **0.50** |

| Ground Terms Only | Use Case | Standard | Match Scheme | Symmetric Effort | | Effort w Edges | Combined | Use Case Average |
|---|---|---|---|---|---|---|---|---|
| | *wo-wo* | 0.84 | 0.75 | 0.89 | 0.88 | 0.87 | 0.72 | **0.83** |
| | *wo-t1* | 0.6 | 0.55 | 0.64 | 0.64 | 0.63 | 0.52 | **0.60** |
| | *wo-t2* | 0.76 | 0.67 | 0.8 | 0.8 | 0.79 | 0.65 | **0.75** |
| | *wo-t3* | 0.82 | 0.75 | 0.87 | 0.86 | 0.86 | 0.71 | **0.81** |
| | *wo-t4* | 0.44 | 0.43 | 0.5 | 0.5 | 0.49 | 0.39 | **0.46** |
| | *wo-t5* | 0.53 | 0.52 | 0.6 | 0.59 | 0.59 | 0.48 | **0.55** |
| | *wo-t6* | 0.67 | 0.65 | 0.76 | 0.75 | 0.73 | 0.58 | **0.69** |
| | *wo-wo EB* | 0.02 | 0.08 | 0.07 | 0.06 | 0.05 | 0.04 | **0.05** |
| **Ground Terms Averages** | | **0.59** | **0.55** | **0.64** | **0.64** | **0.63** | **0.51** | **0.59** |

| (Sibling = Direct) | Use Case | Standard | Match Scheme | Symmetric Effort | | Effort w Edges | Combined | Use Case Average |
|---|---|---|---|---|---|---|---|---|
| | *wo-wo* | 0.73 | 0.64 | 0.76 | 0.76 | 0.75 | 0.61 | **0.71** |
| | *wo-t1* | 0.53 | 0.49 | 0.57 | 0.56 | 0.56 | 0.45 | **0.53** |
| | *wo-t2* | 0.66 | 0.58 | 0.68 | 0.68 | 0.67 | 0.55 | **0.64** |
| | *wo-t3* | 0.63 | 0.58 | 0.67 | 0.67 | 0.66 | 0.55 | **0.63** |
| | *wo-t4* | 0.37 | 0.34 | 0.4 | 0.4 | 0.39 | 0.31 | **0.37** |
| | *wo-t5* | 0.48 | 0.43 | 0.5 | 0.5 | 0.49 | 0.4 | **0.47** |
| | *wo-t6* | 0.57 | 0.54 | 0.63 | 0.62 | 0.6 | 0.48 | **0.57** |
| | *wo-wo EB* | 0.11 | 0.14 | 0.15 | 0.15 | 0.14 | 0.11 | **0.13** |
| **(Sibling=Direct) Averages** | | 0.51 | 0.47 | 0.55 | 0.54 | 0.53 | 0.43 | **0.51** |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Recall Averages** | | **0.53** | **0.50** | **0.57** | **0.57** | **0.56** | **0.46** | **0.53** |

The following two tables contain detailed results for each proximity measure in the Real-Life tests discussed in Chapter 4.

**Table 23. *FutureShop* vs. *BestBuy* Precision.**

| Use Case | Standard | Match Scheme | Symmetric Effort | | Effort w Edges | Combined | Use Case Average |
|---|---|---|---|---|---|---|---|
| Benchmark Averages | 0.14 | 0.55 | 0.78 | 0.83 | 0.83 | 0.45 | **0.60** |
| Ground Term Averages | 0.09 | 0.53 | 0.77 | 0.82 | 0.82 | 0.44 | **0.58** |
| (Sibling=Direct) Averages | 1.00 | 0.55 | 1.00 | 1.00 | 1.00 | 0.55 | **0.85** |
| **Precision Average** | **0.41** | **0.54** | **0.85** | **0.88** | **0.88** | **0.48** | **0.68** |

**Table 24. *FutureShop* vs. *BestBuy* Recall Measures.**

| Use Case | Standard | Match Scheme | Symmetric Effort | | Effort w Edges | Combined | Use Case Average |
|---|---|---|---|---|---|---|---|
| Benchmark Averages | 0.09 | 0.33 | 0.50 | 0.52 | 0.52 | 0.29 | **0.38** |
| Ground Term Averages | 0.05 | 0.28 | 0.41 | 0.44 | 0.44 | 0.23 | **0.31** |
| (Sibling=Direct) Averages | 0.63 | 0.35 | 0.63 | 0.63 | 0.63 | 0.35 | **0.54** |
| **Precision Average** | **0.26** | **0.32** | **0.51** | **0.53** | **0.53** | **0.29** | **0.41** |

# Appendix 2. OWL 2 Representation

## OWL 2 Semantics

The following tables list the OWL 2 expressions utilized by the algorithm presented in this thesis. For ease of readability, the list of notations is presented in Functional Syntax. In each table, the Definition gives the official OWL 2 definition and the Usage column explains how the algorithm uses the expression. For a complete list of OWL 2 syntax and semantics visit the official W3C website contributed to by Horrocks et. al. [78].

**Table 25. Class Expressions**

| Axiom | Definition | Usage |
|---|---|---|
| SubClassOf ( $C_x$ $C_n$ ) | $C_x$ is a sub-class of $C_n$ | Used to represent sub-class, as well as make union of *granules* a sub-class of a concept. |
| EquivalentClasses ( $C_1$ ... $C_n$ ) | All concepts $C_1$ to $C_n$ are equivalent | Used to make two *granules* equivalent before checking whether their *rule* and *data-points* are consistent. |

**Table 26. Interpreting Class Expressions**

| Class Expression | Definition | Usage |
|---|---|---|
| ObjectIntersectionOf ( $C_1$ ... $C_{1n}$ ) | A intersection of concepts $C_1$ *to* $C_n$. The resulting expression is satisfied by all concepts included. | Used to create a Concept out of an intersection of *data-points*, via DataAllValuesFrom() and a single *data-points* via DataHasValue(). |
| ObjectUnionOf ( $C_1$ ... $C_n$ ) | A union of concepts $C_1$ *to* $C_n$. The resulting expression is satisfied by at least one of the concepts included. | Used to group granules together, to be used as a sub-class of a concept. |
| DataAllValuesFrom ( $p_a$ $DR_n$ ) | Assign a data range of literals in $DR_n$ to the property $p_a$. | Creates a Concept expressions out of an range of data literals for a particular property $p_a$. A range is an interpretation of a decision tree rule, such as $p_a \geq l_n$, for some literal $l_n$ |
| DataHasValue ( $p_a$ $l_n$ ) | $p_a = l_n$<br>Assign a single literal value $l_n$ to property $p_a$: | Used to set a *granule*'s *data-points* value $l_n$ for the property $p_a$. |

**Table 27. Data Property Expressions**

| Axiom | Definition | Usage |
|---|---|---|
| EquivalentDataProperty ( $da_1$ ... $da_n$ ) | All data properties $da_1$ to *da* are equivalent | Used to make a local and target data property equivalent. |

**Table 28. Interpreting Data Ranges**

| Data Range | Definition | Usage |
|---|---|---|
| DataOneOf ( $l_1$ ... $l_n$ ) | A set of literal values $l_1$ to $l_n$, such as {"N", "W", "S", "E"} | Used to set a "range" of nominal values, when defining a *rule* with a nominal property. |
| DatatypeRestriction ( $type_a f_1 lt_1$ ... $f_n$ $lt_n$ ) | A set of data ranges of the same type $type_n$, defined by using facets $f$ and literals $l$ . Facets are conditions such as ≥ or ≤. | Used to set a "range" of numeric values, when defining a *rule* with a numeric property. |

## OWL 2 Examples

Examples of OWL 2 syntax usage in the algorithm. For ease of readability the examples are presented in OWL XML syntax.

The rules built have the following form.

**Example 1. A concept as a disjunction of granules (Definition 3).**

| ***Definition*** | $AC_z \equiv (Ag_i \vee Ag_j \vee Ag_k)$ |
|---|---|
| **OWL XML Syntax** | `<EquivalentClasses>`<br>`  <Class IRI="ACz"/>`<br>`  <ObjectUnionOf>`<br>`  <Class IRI="Agi"/>`<br>`  <Class IRI="Agj"/>`<br>`  <Class IRI="Agk"/>`<br>`  </ObjectUnionOf>`<br>`</EquivalentClasses>` |

**Example 2. Granule rule as disjunction of conditions (Definition 4) with numeric $p_a$ and nominal $p_b$ attributes.**

| ***Definition*** | $Ag_i.rule \sqsupseteq [\ (Ap_a \leq l_0) \wedge (Ap_a \geq l_1) \wedge (Ap_b \in \{\ l_3,\ l_4\ ,l_5,l_6\})\ ]$ |
|---|---|
| **OWL XML Syntax** | `<SubClassOf>`<br>`  <Class IRI="Agi"/>`<br>`  <ObjectIntersectionOf>`<br>`  <DataAllValuesFrom>`<br>`  <DataProperty IRI="Apa"/>`<br>`  <DatatypeRestriction>`<br>`  <Datatype abbreviatedIRI="xsd:float"/>`<br>`  <FacetRestriction facet="&xsd;minInclusive">`<br>`  <Literal datatypeIRI="&xsd;float">l0</Literal>`<br>`  </FacetRestriction>`<br>`  </DatatypeRestriction>`<br>`  </DataAllValuesFrom>`<br>`  <DataAllValuesFrom>`<br>`  <DataProperty IRI="Apa"/>`<br>`  <DatatypeRestriction>`<br>`  <Datatype abbreviatedIRI="xsd:float"/>`<br>`  <FacetRestriction facet="&xsd;maxInclusive">`<br>`  <Literal datatypeIRI="&xsd;float">l1</Literal>`<br>`  </FacetRestriction>`<br>`  </DatatypeRestriction>`<br>`  </DataAllValuesFrom>`<br>`  <DataAllValuesFrom>`<br>`  <DataProperty IRI="Apb"/>`<br>`  <DataOneOf>`<br>`  <Literal xml:lang="en">l3</Literal>`<br>`  <Literal xml:lang="en">l4</Literal>`<br>`  <Literal xml:lang="en">l5</Literal>`<br>`  <Literal xml:lang="en">l6</Literal>`<br>`  </DataOneOf>`<br>`  </DataAllValuesFrom>`<br>`  </ObjectIntersectionOf>`<br>`</SubClassOf>` |

**Example 3. A granule rule with a single numeric property (Definition 4).**

| ***Definition*** | $Ag_j.rule \sqsupseteq (Ap_a \leq l_2)$ |
|---|---|
| **OWL XML Syntax** | `<SubClassOf>`<br>`  <Class IRI="Ag`$_j$`"/>`<br>`  <DataAllValuesFrom>`<br>`  <DataProperty IRI="Ap`$_a$`"/>`<br>`  <DatatypeRestriction>`<br>`  <Datatype abbreviatedIRI="xsd:float"/>`<br>`  <FacetRestriction facet="&xsd;minInclusive">`<br>`  <Literal datatypeIRI="&xsd;float">l`$_2$`</Literal>`<br>`  </FacetRestriction>`<br>`  </DatatypeRestriction>`<br>`  </DataAllValuesFrom>`<br>`</SubClassOf>` |

**Example 4. A granule rule with a single nominal property, and multiple values (Definition 4).**

| ***Definition*** | $Ag_k.rule \sqsupseteq (Ap_b \in \{ l_4, l_5, l_6, l_7\})$ |
|---|---|
| **OWL XML Syntax** | `<SubClassOf>`<br>`  <Class IRI="Ag`$_j$`"/>`<br>`  <DataAllValuesFrom>`<br>`  <DataProperty IRI="Ap`$_b$`"/>`<br>`  <DataOneOf>`<br>`  <Literal xml:lang="en">l`$_4$`</Literal>`<br>`  <Literal xml:lang="en">l`$_5$`</Literal>`<br>`  <Literal xml:lang="en">l`$_6$`</Literal>`<br>`  <Literal xml:lang="en">l`$_7$`</Literal>`<br>`  </DataOneOf>`<br>`  </DataAllValuesFrom>`<br>`</SubClassOf>` |

**Example 5. A granule data-points with multiple attributes either numeric or nominal, each having multiple values (Definition 4).**

| ***Definition*** | $Ag_i.data\text{-}points \sqsupseteq [ (Ap_a = l_0) \wedge (Ap_a = l_1) \wedge (Ap_b = l_3 ) \wedge (Ap_b = l_4 ) \wedge (Ap_b = l_5 ) \wedge (Ap_b = l_6 )]$ |
|---|---|
| **OWL XML Syntax** | <SubClassOf><br>  <Class IRI="$Ag_i$"/><br>  <ObjectIntersectionOf><br>  <DataHasValue><br>  <DataProperty IRI="$Ap_a$"/><br>  <Literal datatypeIRI="&xsd;float">$l_0$</Literal><br>  </DataHasValue><br>  <DataHasValue><br>  <DataProperty IRI="$Ap_a$"/><br>  <Literal datatypeIRI="&xsd;float">$l_1$</Literal><br>  </DataHasValue><br>  <DataHasValue><br>  <DataProperty IRI="$Ap_b$"/><br>  <Literal xml:lang="en">$l_3$</Literal><br>  </DataHasValue><br>  <DataHasValue><br>  <DataProperty IRI="$Ap_b$"/><br>  <Literal xml:lang="en">$l_4$</Literal><br>  </DataHasValue><br>  <DataHasValue><br>  <DataProperty IRI="$Ap_b$"/><br>  <Literal xml:lang="en">$l_5$</Literal><br>  </DataHasValue><br>  <DataHasValue><br>  <DataProperty IRI="$Ap_b$"/><br>  <Literal xml:lang="en">$l_6$</Literal><br>  </DataHasValue><br>  </ObjectIntersectionOf><br></SubClassOf> |

**Example 6. A granule data-points with a single property, either numeric or nominal, and a single value (Definition 4).**

| ***Definition*** | $Ag_j.data\text{-}points \sqsupseteq (Ap_a = l_2)$ |
|---|---|
| **OWL XML Syntax** | <SubClassOf><br>  <Class IRI="$Ag_j$"/><br>  <DataHasValue><br>  <DataProperty IRI="$Ap_a$"/><br>  <Literal xml:lang="en">$l_2$</Literal><br>  </DataHasValue><br></SubClassOf> |

**Example 7. A granule data-points with a single nominal property, and multiple values (Definition 4).**

| ***Definition*** | $Ag_k.data\text{-}points \sqsupseteq [(Ap_b = l_3) \wedge (Ap_b = l_4) \wedge (Ap_b = l_5) \wedge (Ap_b = l_6)]$ |
|---|---|
| **OWL XML Syntax** | <SubClassOf><br>  <Class IRI="Ag$_j$"/><br>  <DataHasValue><br>  <DataProperty IRI="Ap$_b$"/><br>  <Literal xml:lang="en">l$_4$</Literal><br>  </DataHasValue><br>  <DataHasValue><br>  <DataProperty IRI="Ap$_b$"/><br>  <Literal xml:lang="en">l$_5$</Literal><br>  </DataHasValue><br>  <DataHasValue><br>  <DataProperty IRI="Ap$_b$"/><br>  <Literal xml:lang="en">l$_6$</Literal><br>  </DataHasValue><br>  <DataHasValue><br>  <DataProperty IRI="Ap$_b$"/><br>  <Literal xml:lang="en">l$_7$</Literal><br>  </DataHasValue><br></SubClassOf> |

**Example 8. Granule Equivalence (Definition 8, line 7).**

| ***Definition*** | $Lg_i \equiv Tg_j$ (based on $Lg_i.data\text{-}points \doteq Tg_j.rule$ as defined above) |
|---|---|
| **OWL XML Syntax** | <EquivalentClasses><br>  <Class IRI="Lg$_i$"/><br>  <Class IRI="Tg$_j$"/><br></EquivalentClasses> |

**Example 9. Data Property Equivalence (Definition 8, line 1).**

| ***Definition*** | *(local)* $Lda_a \equiv_{dp}$ *(target)* $Tda_b$ |
|---|---|
| **OWL XML Syntax** | <EquivalentDataProperties><br>  <DataProperty IRI="Lda$_a$"/><br>  <DataProperty IRI="Tda$_b$"/><br></EquivalentDataProperties> |

# Appendix 3. Use Case Ontologies

The following tables list the mappings between the weather ontology *wo*, and other use cases included in the tests in Chapter 4. These tables also show the correct matches between **Local Concept** and the **Target Concept**. The table also includes comments describing any changes which were done in tables *t1* to *t6*. The formatting of the table is based on Ruby syntax. Comments are prefixed with a (#) pound sign. Strings are enclosed by single quotes (‘). Namespaces have been removed for ease of readability. A match is identified by the Ruby symbol (=>), where ‘A’ => [‘B’] means A in the local ontology matches B in the target ontology. In instances where a single local concept matches multiple target concepts, the relationship is identified by ‘A’ => [‘B’, ‘C’, ‘D’], where A is matched with B, C, and D. A case where a local concept does not have a match in the target ontology, is represented by ‘A’ => []. Table 29 acts as the main references. If one of the subsequent tables does not have a local concept in the left column, this indicates it was not modified, and Table 29 acts as the reference to its unmodified match.

**Table 29. Use Case (*ow* to *ow*): self match.**

| Local Concept | Target Concept |
|---|---|
| 'owl:Thing' | => ['owl:Thing’] |
| 'WeatherDescriptor' | => ['WeatherDescriptor’] |
| 'Partial' | => ['Partial’] |
| 'Showers' | => ['Showers’] |
| 'Thunderstorm' | => ['Thunderstorm’] |
| 'Patches' | => ['Patches’] |
| 'Freezing' | => ['Freezing’] |
| 'Blowing' | => ['Blowing’] |
| 'Shallow' | => ['Shallow’] |
| 'WeatherQualifier' | => ['WeatherQualifier’] |
| 'WeatherIntensity' | => ['WeatherIntensity’] |
| 'WeatherProximity' | => ['WeatherProximity’] |
| 'ReportModifier' | => ['ReportModifier’] |
| 'WeatherEvent' | => ['WeatherEvent’] |
| 'TimedWeatherEvent' | => ['TimedWeatherEvent’] |
| 'WindShiftEvent' | => ['WindShiftEvent’] |
| 'TimeRangeWeatherEvent' | => ['TimeRangeWeatherEvent’] |
| 'DatedWeatherEvent' | => ['DatedWeatherEvent’] |
| 'WeatherReport' | => ['WeatherReport’] |
| 'TafReport' | => ['TafReport’] |
| 'MetarReport' | => ['MetarReport’] |
| 'SpeciReport' | => ['SpeciReport’] |
| 'CanadaMetarReport' | => ['CanadaMetarReport’] |
| 'UsaMetarReport' | => ['UsaMetarReport’] |
| 'IntlMetarReport' | => ['IntlMetarReport’] |

| | |
|---|---|
| 'Measured' | => ['Measured’] |
| 'TemperatureEvent' | => ['TemperatureEvent’] |
| 'CurrentDewPoint' | => ['CurrentDewPoint’] |
| 'MaximumTemperature' | => ['MaximumTemperature’] |
| 'TwentyFourHourMaximumTemp' | => ['TwentyFourHourMaximumTemp’] |
| 'SixHourMaximumTemp' | => ['SixHourMaximumTemp’] |
| 'MinimumTemperature' | => ['MinimumTemperature’] |
| 'SixHourMinimumTemp' | => ['SixHourMinimumTemp’] |
| 'TwentyFourHourMinimumTemp' | => ['TwentyFourHourMinimumTemp’] |
| 'CurrentTemperature' | => ['CurrentTemperature’] |
| 'DerivedTemperature' | => ['DerivedTemperature’] |
| 'WindChill' | => ['WindChill’] |
| 'HeatIndex' | => ['HeatIndex’] |
| 'RelativeHumidity' | => ['RelativeHumidity’] |
| 'PressureEvent' | => ['PressureEvent’] |
| 'PressureTendencyCharacter' | => ['PressureTendencyCharacter’] |
| 'PressureChangeEvent' | => ['PressureChangeEvent’] |
| 'PressureFallingRapidly' | => ['PressureFallingRapidly’] |
| 'PressureRisingRapidly' | => ['PressureRisingRapidly’] |
| 'ThreeHourPressureTendency' | => ['ThreeHourPressureTendency’] |
| 'SeaLevelPressure' | => ['SeaLevelPressure’] |
| 'AltimeterSetting' | => ['AltimeterSetting’] |
| 'SevereWeather' | => ['SevereWeather’] |
| 'TornadicActivity' | => ['TornadicActivity’] |
| 'Tornado' | => ['Tornado’] |
| 'Waterspout' | => ['Waterspout’] |
| 'FunnelCloud' | => ['FunnelCloud’] |
| 'Sandstorm' | => ['Sandstorm’] |
| 'DustOrSandWhirl' | => ['DustOrSandWhirl’] |
| 'Duststorm' | => ['Duststorm’] |
| 'Squall' | => ['Squall’] |
| 'WindEvent' | => ['WindEvent’] |
| 'WindShiftEvent' | => ['WindShiftEvent’] |
| 'VariableWindEvent' | => ['VariableWindEvent’] |
| 'HighVariableWindEvent' | => ['HighVariableWindEvent’] |
| 'LowVariableWindEvent' | => ['LowVariableWindEvent’] |
| 'PeakWindEvent' | => ['PeakWindEvent’] |
| 'GustingWindEvent' | => ['GustingWindEvent’] |
| 'VisibilityEvent' | => ['VisibilityEvent’] |

| | |
|---|---|
| 'SurfaceVisibility' | => ['SurfaceVisibility’] |
| 'RunwayVisualRange' | => ['RunwayVisualRange’] |
| 'UsaRunwayVisualRange' | => ['UsaRunwayVisualRange’] |
| 'IntlRunwayVisualRange' | => ['IntlRunwayVisualRange’] |
| 'SectorVisibility' | => ['SectorVisibility’] |
| 'VerticalVisibility' | => ['VerticalVisibility’] |
| 'UsaSurfaceVisibility' | => ['UsaSurfaceVisibility’] |
| 'IntlSurfaceVisibility' | => ['IntlSurfaceVisibility’] |
| 'SkyCondition' | => ['SkyCondition’] |
| 'CloudLayer' | => ['CloudLayer’] |
| 'SolidOvercastCloudLayer' | => ['SolidOvercastCloudLayer’] |
| 'ScatteredCloudLayer' | => ['ScatteredCloudLayer’] |
| 'FewCloudLayer' | => ['FewCloudLayer’] |
| 'OvercastCloudLayer' | => ['OvercastCloudLayer’] |
| 'BrokenCloudLayer' | => ['BrokenCloudLayer’] |
| 'ClearSkies' | => ['ClearSkies’] |
| 'ClearSkiesSKC' | => ['ClearSkiesSKC’] |
| 'ClearSkiesCLR' | => ['ClearSkiesCLR’] |
| 'TowerVisibility' | => ['TowerVisibility’] |
| 'VariableVisibility' | => ['VariableVisibility’] |
| 'CurrentWeatherEvent' | => ['CurrentWeatherEvent’] |
| 'ObscurationEvent' | => ['ObscurationEvent’] |
| 'VolcanicAsh' | => ['VolcanicAsh’] |
| 'Mist' | => ['Mist’] |
| 'WidespreadDust' | => ['WidespreadDust’] |
| 'Sand' | => ['Sand’] |
| 'Haze' | => ['Haze’] |
| 'Spray' | => ['Spray’] |
| 'Smoke' | => ['Smoke’] |
| 'Fog' | => ['Fog’] |
| 'PrecipitationEvent' | => ['PrecipitationEvent’] |
| 'Drizzle' | => ['Drizzle’] |
| 'Hail' | => ['Hail’] |
| 'Snow' | => ['Snow’] |
| 'IceCrystals' | => ['IceCrystals’] |
| 'Rain' | => ['Rain’] |
| 'SmallHail' | => ['SmallHail’] |
| 'SnowGrains' | => ['SnowGrains’] |
| 'IcePellets' | => ['IcePellets’] |

| | |
|---|---|
| 'CloudType' | => ['CloudType'] |
| 'StandingLenticularOrRotorClouds' | => ['StandingLenticularOrRotorClouds'] |
| 'CumulonimbusMammatus' | => ['CumulonimbusMammatus'] |
| 'ToweringCumulus' | => ['ToweringCumulus'] |
| 'Cumulonimbus' | => ['Cumulonimbus'] |
| 'AltocumulusCastellanus' | => ['AltocumulusCastellanus'] |
| 'OtherWeatherPhenomena' | => ['OtherWeatherPhenomena'] |
| 'UnitAbbreviation' | => ['UnitAbbreviation'] |

**Table 30. Use Case (*ow* to *t1*): flattened concepts.**

| Local concept | Target Concept |
|---|---|
| # flattened DerivedTemperature | |
| 'DerivedTemperature' | => ['WindChill', 'HeatIndex', 'RelativeHumidity'] |
| 'WindChill' | => ['WindChill'] # shifted up to TemperatureEvent |
| 'HeatIndex' | => ['HeatIndex'] # shifted up to TemperatureEvent |
| 'RelativeHumidity' | => ['RelativeHumidity'] # shifted up to TemperatureEvent |
| | |
| # flattened PressureChangeEvent | |
| 'PressureChangeEvent' | => ['PressureFallingRapidly', 'PressureRisingRapidly'] |
| 'PressureFallingRapidly' | => ['PressureFallingRapidly'] # shifted up to PressureEvent |
| 'PressureRisingRapidly' | => ['PressureRisingRapidly'] # shifted up to PressureEvent |
| | |
| # flattened VariableWindEvent | |
| 'VariableWindEvent' | => ['HighVariableWindEvent', 'LowVariableWindEvent'] |
| 'HighVariableWindEvent' | => ['HighVariableWindEvent'] # shifted up to WindEvent |
| 'LowVariableWindEvent' | => ['LowVariableWindEvent'] # shifted up to WindEvent |
| | |
| # flattened CloudLayer | |
| 'CloudLayer' | => ['BrokenCloudLayer', 'FewCloudLayer', 'OvercastCloudLayer', 'ScatteredCloudLayer', 'SolidOvercastCloudLayer'] |
| 'SolidOvercastCloudLayer' | => ['SolidOvercastCloudLayer'] # shifted up to SkyCondition |
| 'ScatteredCloudLayer' | => ['ScatteredCloudLayer'] # shifted up to SkyCondition |
| 'FewCloudLayer' | => ['FewCloudLayer'] # shifted up to SkyCondition |
| 'OvercastCloudLayer' | => ['OvercastCloudLayer'] # shifted up to SkyCondition |
| 'BrokenCloudLayer' | => ['BrokenCloudLayer'] # shifted up to SkyCondition |
| | |
| # flattened ClearSkies | |
| 'ClearSkies' | => ['ClearSkiesCLR', 'ClearSkiesSKC'] |
| 'ClearSkiesSKC' | => ['ClearSkiesSKC'] # shifted up to SkyCondition |

| | |
|---|---|
| 'ClearSkiesCLR' | => ['ClearSkiesCLR’] # shifted up to SkyCondition |

**Table 31. Use Case (*ow* to *t2*): expanded concepts.**

| Local concept | Target Concept |
|---|---|
| # expand WeatherDescriptor | |
| 'WeatherDescriptor' | => ['WeatherDescriptor', 'MinorWeatherDescriptor', 'MajorWeatherDescriptor’] |
| 'Shallow' | => ['Shallow', 'MinorWeatherDescriptor’] # expended under MinorWeatherDescriptor (new) |
| 'Partial' | => ['Partial', 'MinorWeatherDescriptor’] # expended under MinorWeatherDescriptor (new) |
| 'Showers' | => ['Showers', 'MinorWeatherDescriptor’] # expended under MinorWeatherDescriptor (new) |
| 'Patches' | => ['Patches', 'MinorWeatherDescriptor’] # expended under MinorWeatherDescriptor (new) |
| 'Freezing' | => ['Freezing', 'MajorWeatherDescriptor’] # expended under MajorWeatherDescriptor (new) |
| 'Blowing' | => ['Blowing', 'MajorWeatherDescriptor’] # expended under MajorWeatherDescriptor (new) |
| 'Thunderstorm' | => ['Thunderstorm', 'MajorWeatherDescriptor’] # expended under MajorWeatherDescriptor (new) |
| | |
| # expanding ObscurationEvent | |
| 'ObscurationEvent' | => ['ObscurationEvent', 'NonSolidObscurationEvent', 'OtherObscurationEvent', 'SolidObscurationEvent’] |
| 'Mist' | => ['Mist', 'NonSolidObscurationEvent’] # expended under NonSolidObscurationEvent (new) |
| 'Haze' | => ['Haze', 'NonSolidObscurationEvent’] # expended under NonSolidObscurationEvent (new) |
| 'Spray' | => ['Spray', 'NonSolidObscurationEvent’] # expended under NonSolidObscurationEvent (new) |
| 'Fog' | => ['Fog', 'NonSolidObscurationEvent’] # expended under NonSolidObscurationEvent (new) |
| 'VolcanicAsh' | => ['VolcanicAsh', 'SolidObscurationEvent’] # expended under SolidObscurationEvent (new) |
| 'WidespreadDust' | => ['WidespreadDust', 'SolidObscurationEvent’] # expended under SolidObscurationEvent (new) |
| 'Sand' | => ['Sand', 'SolidObscurationEvent’] # expended under SolidObscurationEvent (new) |
| 'Smoke' | => ['Smoke', 'OtherObscurationEvent’] # expended under OtherObscurationEvent (new) |
| | |
| # expanded PrecipitationEvent | |
| 'PrecipitationEvent' | => ['PrecipitationEvent', 'SolidPrecipitationEvent', 'LiquidPrecipitationEvent’] |

| | |
|---|---|
| 'Drizzle' | => ['Drizzle', 'LiquidPrecipitationEvent’]<br># expended under LiquidPrecipitationEvent (new) |
| 'Rain' | => ['Rain', 'LiquidPrecipitationEvent’]<br># expended under LiquidPrecipitationEvent (new) |
| 'Hail' | => ['Hail', 'SolidPrecipitationEvent’]<br># expended under SolidPrecipitationEvent (new) |
| 'Snow' | => ['Snow', 'SolidPrecipitationEvent’]<br># expended under SolidPrecipitationEvent (new) |
| 'IceCrystals' | => ['IceCrystals', 'SolidPrecipitationEvent’]<br># expended under SolidPrecipitationEvent (new) |
| 'SmallHail' | => ['SmallHail', 'SolidPrecipitationEvent’]<br># expended under SolidPrecipitationEvent (new) |
| 'SnowGrains' | => ['SnowGrains', 'SolidPrecipitationEvent’]<br># expended under SolidPrecipitationEvent (new) |
| 'IcePellets' | => ['IcePellets', 'SolidPrecipitationEvent’]<br># expended under SolidPrecipitationEvent (new) |

**Table 32. Use Case (*ow* to *t3*): drastic reconstruction.**

| **Local concept** | **Target Concept** |
|---|---|
| # moved WeatherDescriptor | |
| 'WeatherDescriptor' | => ['WeatherDescriptor’] # moved under WeatherQualifier |
| 'Partial' | => ['Partial’] # moved with WeatherQualifier |
| 'Showers' | => ['Showers’] # moved with WeatherQualifier |
| 'Thunderstorm' | => ['Thunderstorm’] # moved with WeatherQualifier |
| 'Patches' | => ['Patches’] # moved with WeatherQualifier |
| 'Freezing' | => ['Freezing’] # moved with WeatherQualifier |
| 'Blowing' | => ['Blowing’] # moved with WeatherQualifier |
| 'Shallow' | => ['Shallow’] # moved with WeatherQualifier |
| | |
| # removed Measured | |
| 'Measured' | => ['TemperatureEvent', 'PressureEvent’] |
| 'TemperatureEvent' | => ['TemperatureEvent’] # moved to CurrentWeatherEvent |
| 'CurrentDewPoint' | => ['CurrentDewPoint’] # moved with TemperatureEvent |
| 'MaximumTemperature' | => ['MaximumTemperature’] # moved with TemperatureEvent |
| 'TwentyFourHourMaximumTemp' | => ['TwentyFourHourMaximumTemp’] # moved with MaximumTemperature |
| 'SixHourMaximumTemp' | => ['SixHourMaximumTemp’] # moved with MaximumTemperature |
| 'MinimumTemperature' | => ['MinimumTemperature’] # moved with TemperatureEvent |
| 'SixHourMinimumTemp' | => ['SixHourMinimumTemp’] # moved with MinimumTemperature |
| 'TwentyFourHourMinimumTemp' | => ['TwentyFourHourMinimumTemp’] # moved with MinimumTemperature |

| | |
|---|---|
| 'CurrentTemperature' | => ['CurrentTemperature’] # moved with TemperatureEvent |
| 'DerivedTemperature' | => ['DerivedTemperature’] # moved with TemperatureEvent |
| 'WindChill' | => ['WindChill’] # moved with DerivedTemperature |
| 'HeatIndex' | => ['HeatIndex’] # moved with DerivedTemperature |
| 'RelativeHumidity' | => ['RelativeHumidity’] # moved with DerivedTemperature |

| | |
|---|---|
| # removed PressureEvent | |
| 'PressureEvent' | => ['PressureEvent’] # moved under UnitAbbreviation |
| 'PressureTendencyCharacter' | => ['PressureTendencyCharacter’] # moved with PressureEvent |
| 'PressureChangeEvent' | => ['PressureChangeEvent’] # moved with PressureEvent |
| 'PressureFallingRapidly' | => ['PressureFallingRapidly’] # moved with PressureChangeEvent |
| 'PressureRisingRapidly' | => ['PressureRisingRapidly’] # moved with PressureChangeEvent |
| 'ThreeHourPressureTendency' | => ['ThreeHourPressureTendency’] # moved with PressureEvent |
| 'SeaLevelPressure' | => ['SeaLevelPressure’] # moved with PressureEvent |
| 'AltimeterSetting' | => ['AltimeterSetting’] # moved with PressureEvent |

| | |
|---|---|
| # moved CloudLayer | |
| 'CloudLayer' | => ['CloudLayer’] # moved under CloudQualifier (new under WeatherQualifier) |
| 'SolidOvercastCloudLayer' | => ['SolidOvercastCloudLayer’] # moved with CloudLayer |
| 'ScatteredCloudLayer' | => ['ScatteredCloudLayer’] # moved with CloudLayer |
| 'FewCloudLayer' | => ['FewCloudLayer’] # moved with CloudLayer |
| 'OvercastCloudLayer' | => ['OvercastCloudLayer’] # moved with CloudLayer |
| 'BrokenCloudLayer' | => ['BrokenCloudLayer’] # moved with CloudLayer |

| | |
|---|---|
| # moved CloudType | |
| 'CloudType' | => ['CloudType’] # moved under CloudQualifier (new under WeatherQualifier) |
| 'StandingLenticularOrRotorClouds' | => ['StandingLenticularOrRotorClouds’] # moved with CloudType |
| 'CumulonimbusMammatus' | => ['CumulonimbusMammatus’] # moved with CloudType |
| 'ToweringCumulus' | => ['ToweringCumulus’] # moved with CloudType |
| 'Cumulonimbus' | => ['Cumulonimbus’] # moved with CloudType |
| 'AltocumulusCastellanus' | => ['AltocumulusCastellanus’] # moved with CloudType |

**Table 33. Use Case (*ow* to *t4*): ground term movement.**

| Local concept | Target Concept |
|---|---|
| # concepts removed; sub-classes moved under MinorWeatherEvent(new), MediumWeatherEvent(new), MajorWeatherEvent(new), | |
| 'ObscurationEvent' | => [], # ObscurationEvent (deleted) |
| 'VolcanicAsh' | => ['VolcanicAsh'] # moved under MajorWeatherEvent (new) |
| 'Mist' | => ['Mist'] # moved under MinorWeatherEvent (new) |
| 'WidespreadDust' | => ['WidespreadDust'] # moved under MajorWeatherEvent (new) |
| 'Sand' | => ['Sand'] # moved under MediumWeatherEvent (new) |
| 'Haze' | => ['Haze'] # moved under MinorWeatherEvent (new) |
| 'Spray' | => ['Spray'] # moved under MinorWeatherEvent (new) |
| 'Smoke' | => ['Smoke'] # moved under MajorWeatherEvent (new) |
| 'Fog' | => ['Fog'] # moved under MediumWeatherEvent (new) |
| | |
| # moved sub-classes and removed PrecipitationEvent | |
| 'PrecipitationEvent' | => [], # PrecipitationEvent (deleted) |
| 'SnowGrains' | => ['SnowGrains'] # moved under MinorWeatherEvent (new) |
| 'Drizzle' | => ['Drizzle'] # moved under MinorWeatherEvent (new) |
| 'SmallHail' | => ['SmallHail'] # moved under MinorWeatherEvent (new) |
| 'Hail' | => ['Hail'] # moved under MediumWeatherEvent (new) |
| 'Rain' | => ['Rain'] # moved under MediumWeatherEvent (new) |
| 'IcePellets' | => ['IcePellets'] # moved under MajorWeatherEvent (new) |
| 'Snow' | => ['Snow'] # moved under MajorWeatherEvent (new) |
| 'IceCrystals' | => ['IceCrystals'] # moved under MajorWeatherEvent (new) |
| | |
| # moved sub-classes and removed CloudType | |
| 'CloudType' | => [], # deleted CloudTyped |
| 'StandingLenticularOrRotorClouds' | => ['StandingLenticularOrRotorClouds'] # moved under MinorWeatherEvent (new) |
| 'ToweringCumulus' | => ['ToweringCumulus'] # moved under MinorWeatherEvent (new) |
| 'Cumulonimbus' | => ['Cumulonimbus'] # moved under MediumWeatherEvent (new) |
| 'AltocumulusCastellanus' | => ['AltocumulusCastellanus'] # moved under MediumWeatherEvent (new) |
| 'CumulonimbusMammatus' | => ['CumulonimbusMammatus'] # moved under MajorWeatherEvent (new) |
| | |
| # removed WeatherDescriptor; sub-classes moved under MiscWeatherEvent(new); MiscVisibilityEvent(new) | |
| 'WeatherDescriptor' | => [], # deleted WeatherDescriptor |
| 'Partial' | => ['Partial'] # moved under MiscWeatherEvent (new) |

| | |
|---|---|
| 'Showers' | => ['Showers’] # moved under MiscVisibilityEvent (new) |
| 'Thunderstorm' | => ['Thunderstorm’] # moved under MiscVisibilityEvent (new) |
| 'Patches' | => ['Patches’] # moved under MiscVisibilityEvent (new) |
| 'Freezing' | => ['Freezing’] # moved under MiscVisibilityEvent (new) |
| 'Blowing' | => ['Blowing’] # moved under MiscWeatherEvent (new) |
| 'Shallow' | => ['Shallow’] # moved under MiscWeatherEvent (new) |

| | |
|---|---|
| # removed SevereWeather; sub-classes moved under MiscWeatherEvent(new); MiscVisibilityEvent(new); | |
| 'SevereWeather' | => [], # deleted SevereWeather |
| 'TornadicActivity' | => ['TornadicActivity’] # moved under MiscVisibilityEvent (new) |
| 'Tornado' | => ['Tornado’] # moved with TornadicActivity |
| 'Waterspout' | => ['Waterspout’] # moved with TornadicActivity |
| 'FunnelCloud' | => ['FunnelCloud’] # moved with TornadicActivity |
| 'Sandstorm' | => ['Sandstorm’] # moved under MiscWeatherEvent (new) |
| 'DustOrSandWhirl' | => ['DustOrSandWhirl’] # moved under MiscVisibilityEvent (new) |
| 'Duststorm' | => ['Duststorm’] # moved under MiscVisibilityEvent (new) |
| 'Squall' | => ['Squall’] # moved under MiscWeatherEvent (new) |

**Table 34. Use Case (*ow* to *t5*) : ground terms become super-classes of other ground terms.**

| **Local concept** | **Target Concept** |
|---|---|
| # restructured WeatherDescriptor ground terms | |
| 'WeatherDescriptor' | => ['WeatherDescriptor’] |
| | |
| 'Blowing' | => ['Blowing’] # became a super-class |
| 'Partial' | => ['Partial’] # moved under Blowing |
| 'Freezing' | => ['Freezing’] # moved under Blowing |
| | |
| 'Patches' | => ['Patches’] # became a super-class |
| 'Showers' | => ['Showers’] # moved under Patches |
| 'Shallow' | => ['Shallow’] # moved under Patches |
| 'Thunderstorm' | => ['Thunderstorm’] # moved under Patches |
| | |
| 'ClearSkiesCLR' | => ['ClearSkiesCLR’] # became a super-class |
| 'GustingWindEvent' | => ['GustingWindEvent’] # became a super-class |
| | |
| # sub-classes (ground terms) moves elsewhere | |
| 'PressureEvent' | => [], # PressureEvent(deleted) |

| | |
|---|---|
| 'PressureTendencyCharacter' | => ['PressureTendencyCharacter’] # moved under ClearSkiesCLR |
| 'ThreeHourPressureTendency' | => ['ThreeHourPressureTendency’] # moved under ClearSkiesCLR |
| | |
| 'SeaLevelPressure' | => ['SeaLevelPressure’] # moved under GustingWindEvent |
| 'AltimeterSetting' | => ['AltimeterSetting’] # moved under GustingWindEvent |
| 'PressureChangeEvent' | => ['PressureChangeEvent’] # moved under GustingWindEvent |
| | |
| 'PressureFallingRapidly' | => ['PressureFallingRapidly’] # moved with PressureChangeEvent |
| 'PressureRisingRapidly' | => ['PressureRisingRapidly’] # moved with PressureChangeEvent |
| | |
| | |
| # sub-classes (ground terms removed) | |
| 'Measured' | => ['Measured’] #kept TemperatureEvent wth MaximumTemperature,MinimumTemperature |
| | |
| 'TemperatureEvent' | => ['TemperatureEvent’] # stayed under Measured |
| 'MaximumTemperature' | => ['MaximumTemperature’] # stayed under TemperatureEvent |
| 'TwentyFourHourMaximumTemp' | => ['TwentyFourHourMaximumTemp’] # stayed under TemperatureEvent |
| 'SixHourMaximumTemp' | => ['SixHourMaximumTemp’] # stayed under TemperatureEvent |
| 'MinimumTemperature' | => ['MinimumTemperature’] # stayed under TemperatureEvent |
| 'SixHourMinimumTemp' | => ['SixHourMinimumTemp’] # stayed under TemperatureEvent |
| 'TwentyFourHourMinimumTemp' | => ['TwentyFourHourMinimumTemp’] # stayed under TemperatureEvent |
| | |
| 'CurrentDewPoint' | => ['CurrentDewPoint’] # moved under ClearSkiesCLR |
| 'CurrentTemperature' | => ['CurrentTemperature’] # moved under GustingWindEvent |
| 'DerivedTemperature' | => ['DerivedTemperature’] # moved under GustingWindEvent |
| 'WindChill' | => ['WindChill’] # moved with DerivedTemperature |
| 'HeatIndex' | => ['HeatIndex’] # moved with DerivedTemperature |
| 'RelativeHumidity' | => ['RelativeHumidity’] # moved with DerivedTemperature |
| | |
| # sub-calss (ground terms) moved elsewhere | |
| 'SevereWeather' | => [], # deleted SevereWeather |
| | |
| 'Sandstorm' | => ['Sandstorm’] # moved under FewCloudLayer |
| 'DustOrSandWhirl' | => ['DustOrSandWhirl’] # moved under FewCloudLayer |
| | |
| 'Duststorm' | => ['Duststorm’] # moved under BrokenCloudLayer |
| 'Squall' | => ['Squall’] # moved under BrokenCloudLayer |

| | |
|---|---|
| 'TornadicActivity' | => ['TornadicActivity'] # moved under BrokenCloudLayer |

| | |
|---|---|
| 'Tornado' | => ['Tornado'] # moved with TornadicActivity |
| 'Waterspout' | => ['Waterspout'] # moved with TornadicActivity |
| 'FunnelCloud' | => ['FunnelCloud'] # moved with TornadicActivity |

**Table 35. Use Case (*ow* to *t6)*: ground terms become super-classes of their siblings.**

| **Local concept** | **Target Concept** |
|---|---|
| # ground-terms moved down under AltocumulusCastellanus | |
| 'CloudType' | => ['CloudType'] |
| 'AltocumulusCastellanus' | => ['AltocumulusCastellanus'] # became super-class |
| 'StandingLenticularOrRotorClouds' | => ['StandingLenticularOrRotorClouds'] # moved under AltocumulusCastellanus |
| 'CumulonimbusMammatus' | => ['CumulonimbusMammatus'] # moved under AltocumulusCastellanus |
| 'ToweringCumulus' | => ['ToweringCumulus'] # moved under AltocumulusCastellanus |
| 'Cumulonimbus' | => ['Cumulonimbus'] # moved under AltocumulusCastellanus |

| | |
|---|---|
| # ground terms became sub-classes | |
| 'ObscurationEvent' | => ['ObscurationEvent'] |
| 'WidespreadDust' | => ['WidespreadDust'] # became a super-class |
| 'VolcanicAsh' | => ['VolcanicAsh'] # moved under WidespreadDust |
| 'Sand' | => ['Sand'] # moved under WidespreadDust |

| | |
|---|---|
| 'Fog' | => ['Fog'] # became a super-class |
| 'Haze' | => ['Haze'] # moved under Fog |
| 'Mist' | => ['Mist'] # moved under Fog |
| 'Smoke' | => ['Smoke'] # moved under Fog |
| 'Spray' | => ['Spray'] # moved under Fog |

| | |
|---|---|
| # ground terms became sub-classes of siblings | |
| 'PrecipitationEvent' | => ['PrecipitationEvent'] |
| 'Snow' | => ['Snow'] # became a super-class |
| 'IceCrystals' | => ['IceCrystals'] # moved under Snow |
| 'SnowGrains' | => ['SnowGrains'] # moved under Snow |
| 'IcePellets' | => ['IcePellets'] # moved under Snow |

| | |
|---|---|
| 'Rain' | => ['Rain'] # became a super-class |
| 'Drizzle' | => ['Drizzle'] # moved under Rain |

| | |
|---|---|
| 'Hail' | => ['Hail'] # moved under Rain |
| 'SmallHail' | => ['SmallHail'] # moved under Rain |

| | |
|---|---|
| # ground terms became sub-classes of siblings | |
| 'WeatherDescriptor' | => ['WeatherDescriptor'] |
| 'Patches' | => ['Patches'] # became a super-class |
| 'Showers' | => ['Showers'] # moved under Patches |
| 'Thunderstorm' | => ['Thunderstorm'] # moved under Patches |
| 'Shallow' | => ['Shallow'] # moved under Patches |

| | |
|---|---|
| 'Blowing' | => ['Blowing'] # became a super-class |
| 'Partial' | => ['Partial'] # moved under Blowing |
| 'Freezing' | => ['Freezing'] # moved under Blowing |

**Table 36. Use Case (*FutureShop* to *BestBuy)*: real-world example.**

| Local concept | Target Concept |
|---|---|
| 'Computers-1' | => ['Computers-1'], |
| 'AppleiPadAppleTabletComputer-2' | => ['AppleMacBooksiMacsiPads-2'], |
| 'AppleComputers-2' | => ['AppleMacBooksiMacsiPads-2'], |
| 'MacMiniiMacComputers-2' | => ['AppleMacBooksiMacsiPads-2'], |
| 'MacBookMacBookProAir-2' | => ['AppleMacBooksiMacsiPads-2'], |
| 'Laptops-2' | => ['Laptops-2', '17Laptops-3', '16Laptops-3'], |
| 'Netbooks-2' | => ['Laptops-2'], |
| 'LaptopAccessories-2' | => ['LaptopAccessories-3'], |
| 'DesktopComputers-2' | => ['DesktopComputers-2'], |
| 'DesktopComputerAccessories-2' | => ['Accessories-2'], |
| 'Monitors-2' | => ['Monitors-2', 'LEDMonitors-3'], |
| 'PrintersandAllInOnes-2' | => ['PrintersInkPaper-2'], |
| 'InkToner-2' | => ['InkToner-3', 'Toner-4'], |
| 'Paper-2' | => ['PaperSupplies-3'], |
| 'DataProjectors-2' | => ['Projectors-2', 'ReplacementLamps-4', 'MiscellaneousAccessories-4'], |
| 'Scanners-2' | => ['Scanners-2'], |
| # Ground Terms | |
| 'iPad-3' | => ['AppleiPad-3'], |
| 'AppleiPadAccessories-3' | => ['AppleiPadAccessories-3', 'iPadCasesCovers-4', 'iPadCablesChargers-4', |
| 'AppleComputers-3' | => ['AppleiMac-3', 'AppleMacMini-3'], |
| 'AppleLaptop-3' | => ['AppleMacBookPro-3', 'AppleMacBookAir-3', 'AppleMacBook-3'], |
| 'AppleComputerAccessories-3' | => ['AppleMonitors-3', 'AppleAccessories-3', 'OtherAppleAccessories-4','AppleKeyboardsMice- |

| | |
|---|---|
| | 4','AppleWirelessBaseStations-4', |
| '13LaptopsandSmaller-3' | => ['141LaptopsUnder-3'], |
| '14Laptops-3' | => ['141LaptopsUnder-3'], |
| '15Laptops-3' | => ['156Laptops-3', '154Laptops-3'], |
| '18LaptopsandLarger-3' | => ['18LaptopsOver-3'], |
| 'RefurbishedLaptops-3' | => ['Laptops-2'], |
| 'TabletandSpeciality-3' | => ['TabletsConvertiblePCs-3'], |
| '10NetbooksandLarger-3' | => ['Netbooks-3'], |
| 'LaptopBagsCases-3' | => ['LaptopCases-4','Sleeves-5','StandardCases-5','MessengerBags-5','Backpacks-5','DesignerCases-5','RollerCases-5','TravelCases-5'], |
| 'LaptopSkins-3' | => ['LaptopCases-4'], |
| 'Batteries-3' | => ['LaptopBatteriesPower-4'], |
| 'DockingStations-3' | => ['DockingStationsStands-4'], |
| 'Locks-3' | => ['Accessories-2'], |
| 'PowerProducts-3' | => ['LaptopBatteriesPower-4'], |
| 'PresentingTools-3' | => ['Accessories-2'], |
| 'ScreenAccessories-3' | => ['Accessories-2'], |
| 'LaptopCooling-3' | => ['LaptopCooling-4'], |
| 'SupportsandStands-3' | => ['Accessories-2'], |
| 'OtherLaptopAccessories-3' | => ['OtherLaptopAccessories-4'], |
| 'DesktopComputers-3' | => ['DesktopComputers-3','DesktopComputerPackages-3'], |
| 'PerformanceGamingComputers-3' | => ['DesktopComputers-3','DesktopComputerPackages-3'], |
| 'AllinOneComputers-3' | => ['AllinOneDesktopComputers-3'], |
| 'HomeServers-3' | => ['HomeServers-3'], |
| 'Adapters-3' | => ['Adapters-5', 'Miscellaneous-4'], |
| 'BatteryBackUp-3' | => ['UPSBackup-5'], |
| 'Bluetooth-3' | => ['Miscellaneous-4'], |
| 'ExtensionCables-3' | => ['MiceKeyboardCables-5'], |
| 'FirewireCables-3' | => ['FireWireCables-5'], |
| 'Headsets-3' | => ['HeadsetsMics-3'], |
| 'MonitorStandsandDeskMounts-3' | => ['MonitorStands-4'], |
| 'MonitorCables-3' | => ['MonitorVideoCables-5'], |
| 'Mousepads-3' | => ['MouseWristPads-4'], |
| 'OtherCables-3' | => ['PowerCables-5'], |
| 'SurgeProtectors-3' | => ['SurgeProtection-4', 'SurgeProtectors-5'], |
| 'TransferCables-3' | => ['SerialATAFloppyCables-5'], |
| 'USBCables-3' | => ['USBCables-5'], |
| 'OtherComputerAccessories-3' | => ['KVMSwitches-5', 'HighDefinitionWebcams-4', 'StandardDefinitionWebcams-4', 'ToolKits-4', 'OtherAccessories-3', 'VideoCapture-3', 'SoundCards-3', 'SerialParallelCards-3', 'Miscellaneous-4'], |
| '19UnderMonitors-3' | => ['19UnderLCDMonitors-3'], |
| '2021Monitors-3' | => ['20215LCDMonitors-3'], |
| '2223Monitors-3' | => ['22235LCDMonitors-3'], |
| '24UpMonitors-3' | => ['24UpLCDMonitors-3'], |
| 'AllinOneInkjetPrinters-3' | => ['InkjetPrinters-3'], |
| 'AllinOneLaserPrinters-3' | => ['AllInOneLaserPrinters-4'], |
| 'PhotoPrinters-3' | => ['PhotoPrinters-4'], |

| | |
|---|---|
| 'LaserPrinters-3' | => ['LaserPrinters-3', 'SingleFunctionLaserPrinters-4'], |
| 'InkRefillKits-3' | => ['RefillKits-4'], |
| 'Brother-3' | => ['InkforBrotherPrinters-4'], |
| 'Canon-3' | => ['InkforCanonPrinters-4'], |
| 'Epson-3' | => ['InkforEpsonPrinters-4'], |
| 'HewlettPackard-3' | => ['InkforHPPrinters-4'], |
| 'Kodak-3' | => ['InkforOtherPrinters-4'], |
| 'Lexmark-3' | => ['InkforLexmarkPrinters-4'], |
| 'Minolta-3' | => ['InkforOtherPrinters-4'], |
| 'NEC-3' | => ['InkforOtherPrinters-4'], |
| 'Okidata-3' | => ['InkforOtherPrinters-4'], |
| 'Panasonic-3' | => ['InkforOtherPrinters-4'], |
| 'Primera-3' | => ['InkforOtherPrinters-4'], |
| 'Samsung-3' | => ['InkforOtherPrinters-4'], |
| 'Sharp-3' | => ['InkforOtherPrinters-4'], |
| 'Xerox-3' | => ['InkforOtherPrinters-4'], |
| 'FaxPaper-3' | => ['PaperSupplies-3'], |
| 'Labels-3' | => ['LabelsStickers-4'], |
| 'MultipurposePaper-3' | => ['MultipurposePaper-4'], |
| 'OtherPaper-3' | => ['PaperMiscellaneous-4'], |
| 'PhotoPaper-3' | => ['PhotoQualityPaper-4'], |
| 'DataProjectors-3' | => ['Projectors-3'], |
| 'PocketProjectors-3' | => ['Projectors-3'], |
| 'BusinessDocumentScanners-3' | => ['BusinessProfessionalScanning-3'], |
| 'PhotoPersonalScanners-3' | => ['HomeScanning-3'] |

# Appendix 4. Code

All the code and supplementary files can be found at:
http://www.scs.ryerson.ca/~bgajdero/msc_thesis

# Appendix 5. User Interface: Match Selector

The appendix gives a sample output of the matches proposed by the algorithm. Each example is accompanied by a comment and its purpose. The results have the following format:

```
Level [x]
# the following matches are for local concepts at level x.

       [y] >> LC1
       # Local concept at level x

       (a) [b] >> TC1
       # LC1 was matched to TC1 using matching scheme (a); TC1 is at level b, where
       LevelT(1) = b;
       # Matching Schemes: 1 = GP, 2 = G2PM, 3 = CM, 4 = CVPM

       (Rating c, Method: X)
       # match rating is c, name of match scheme is X

       Optional information, dependent on type of match

       :: direct
       # a direct match between LC1 and TC1

       :: sibling
       # LC1 was matched to TC1, where TC1 ≈ LC2, LC1 ⇑ LC2

       :: super2sub
       # LC1 was matched to TC1, where TC1 ≈ LC2, LC1 ⊑ LC2

       :: sub2super
       # LC1 was matched to TC1, where TC1 ≈ LC2, LC1 ⊒ LC2

       :: L:[LC1] ⊒ (c10) TC10 ⊒ (c11) TC11 ⊒ (c12) TC12
       # LC1 also has a CM match with TC10, TC11, and TC12, where TC1 ⊒{TC10, TC11, and
       TC12}

       :: T:[LC1] ⊒ (c10) TC10 ⊒ (c11) TC11 ⊒ (c12) TC12
       # same as above, except match was found in the target ontology. Needed when
       finding an indirect match between a local and target class, through a target
       concept which does not exists in the local ontology.
```

**Table 37. Sample of proposed matches presented to the user.**

| **Level [2]** |
|---|
| |
| **[2] >> VisibilityEvent** |
| (1) [4] >> AltimeterSetting<br>(Ranking 0.9706, Method: GM)<br>:: L:[VisibilityEvent] $\sqsupseteq$ (0.9091) BrokenCloudLayer |
| (3) [3] >> LowVariableWindEvent<br>(Ranking 0.9412, Method: CM)<br>:: L:[VisibilityEvent] $\sqsupseteq$ (0.9697) SurfaceVisibility $\sqsupseteq$ (0.9118) TowerVisibility |
| (3) [4] >> PressureEvent<br>(Ranking 0.9167, Method: CM)<br>:: L:[VisibilityEvent] $\sqsupseteq$ (0.9167) SkyCondition $\sqsupseteq$ (0.9167) SurfaceVisibility |
| (3) [4] >> ClearSkies<br>(Ranking 0.9118, Method: CM)<br>:: [L] super2sub<br>:: L:[VisibilityEvent] $\sqsupseteq$ (0.9697) ClearSkies $\sqsupseteq$ (0.9394) SurfaceVisibility |
| (3) [5] >> RelativeHumidity<br>(Ranking 0.9000, Method: CM)<br>:: L:[VisibilityEvent] $\sqsupseteq$ (0.9000) SurfaceVisibility $\sqsupseteq$ (0.9000) SkyCondition |
| (4) [2] >> VisibilityEvent<br>(Ranking 0.9000, Method: CVPM)<br>:: [L] direct |
| |
| **[2] >> Freezing** |
| (1) [4] >> CloudLayer<br>(Ranking 0.9706, Method: GM) |
| (1) [4] >> Freezing<br>(Ranking 0.9412, Method: GM)<br>:: [L] direct |
| (2) [3] >> PressureEvent<br>(Ranking 0.9167, Method: G2PM) |
| (3) [4] >> Fog<br>(Ranking 0.9118, Method: CM) |
| (3) [2] >> Shallow<br>(Ranking 0.9000, Method: CM)<br>:: [L] sibling |
| (4) [5] >> RelativeHumidity<br>(Ranking 0.9000, Method: CVPM) |
| |
| **[2] >> SevereWeather** |
| (3) [4] >> CurrentTemperature<br>(Ranking 0.9706, Method: CM)<br>:: L:[SevereWeather] $\sqsupseteq$ (0.9706) TornadicActivity $\sqsupseteq$ (0.9706) DustOrSandWhirl |
| (3) [4] >> PressureChangeEvent<br>(Ranking 0.9412, Method: CM) |
| (2) [3] >> PressureEvent<br>(Ranking 0.9167, Method: CM)<br>:: L:[SevereWeather] $\sqsupseteq$ (0.9167) Duststorm $\sqsupseteq$ (0.9167) Squall |
| (3) [2] >> CurrentWeatherEvent<br>(Ranking 0.9000, Method: CM)<br>:: [L] sibling<br>:: L:[SevereWeather] $\sqsupseteq$ (0.9000) DustOrSandWhirl $\sqsupseteq$ (0.9000) TornadicActivity |
| (4) [4] >> HighVariableWindEvent<br>(Ranking 0.9118, Method: CVPM)<br>:: L:[SevereWeather] $\sqsupseteq$ (0.9118) Squall |
| |
| |

**Level [3]**

**[3] >> PeakWindEvent**

(3) [4] >> ClearSkies
(Ranking 0.9697, Method: CM)

(3) [3] >> PeakWindEvent
(Ranking 0.9167, Method: CM)
:: [L] direct

**[3] >> Squall**

(1) [4] >> ClearSkies
(Ranking 0.9706, Method: GM)

(1) [4] >> LowVariableWindEvent
(Ranking 0.9412, Method: GM)

(3) [2] >> Shallow
(Ranking 0.9000, Method: CM)
:: [T] sibling

(3) [5] >> RelativeHumidity
(Ranking 0.9000, Method: CM)

# Bibliography


[1] A. Prencipe, A. Davies, and M. Hobday, eds., Introduction. *The Business of Systems Integration*, By Prencipe, Oxford University Press, 2003.

[2] C.P. Cheng, G.T. Lau, and K.H. Law, "Mapping regulations to industry-specific taxonomies," in *Proc. 11th International Conference on Artificial Intelligence and Law*, ACM, 2007, pp. 59-63.

[3] W.S. Li, C. Clifton, and S.Y. Liu, "Database integration using neural networks: implementation and experiences," *Knowledge Information Systems*, Springer-Verang, vol. 2, no. 1, 2000, pp. 73-96.

[4] F. Baader, A. Borgida, and D.L. McGuinness, "Matching in Description Logics: Preliminary Results," in *Proc. 6th International Conference on Conceptual Structures,* (ICCS '98), Springer-Verlag, 1998, pp. 15-34.

[5] V. Papataxiarhis, V. Tsetsos, I. Karali, and P. Stamatopoulos, " Developing rule-based applications for the Web: Methodologies and Tools ", in *Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches*, A. Giurca, D. Gasevic, and K. Taveter, eds., Information Science Reference, 2009.

[6] M. Uschold, and M. Gruninger, "Ontologies: principles, methods, and applications". *Knowledge Engineering Review*, vol. 11 no. 2, 1996, pp. 93-155.

[7] F. Baader, I. Horrocks, and U. Sattler, "Description Logics" in *Handbook of Knowledge Representation*, V. L. Frank van Harmelen and B. Porter eds., Elsevier, 2008, pp. 135-179.

[8] S.M. Falconer, N.F. Noy, and M.A. Storey, "Ontology mapping - a user survey," in *Proc. Workshop on Ontology Matching* (OM2007), 2007, pp. 113-125.

[9] D. Richards, and S.J. Simoff, "Design ontology in context - a situated cognition approach to conceptual modeling". *Artificial Intelligence in Engineering,* vol. 15 no. 2, 2001, pp. 121-136.

[10] J. Euzenat, and P. Shvaiko, eds. *Ontology Matching*. Secaucus, NJ, USA: Springer-Verlag New York, 2007.

[11] J. Hayes, and C. Gutierrez, "Bipartite Graphs as Intermediate Model for RDF," in *Lecture Notes in Computer Science*, LNCS 3298, Springer, 2004, pp. 47-61.

[12] B. Gajderowicz, A. Sadeghian, and M.F. dos Santos, "Expectation Maximization Enhancement with Evolution Strategy for Stochastic Ontology Mapping," in *Proc 11th Annual Conference on Genetic and Evolutionary Computation* (GECCO09), ACM, 2009, pp. 1847-1848.

[13] T. Slimani, B. Yaghlane, and K. Mellouli, "A New Similarity Measure based on Edge Counting," in *Proc. World Academy of Science, Engineering and Technology*, vol. 17, Dec. 2006.

[14] N. Lin, B. Wu, R. Jansen, M. Gerstein, and H. Zhao, "Information assessment on predicting protein-protein interactions". *BMC Bioinformatics*, vol. 5 no. 154, 2004.

[15] P. Doshi, and C. Thomas, "Inexact matching of ontology graphs using expectation maximization," in *Proc. 21st National Conference on Artificial Intelligence* (AAAI'06), AAAI, 2006, pp. 1277-1282.

[16] B. Luo, and E.R. Hancock, "Structural Graph Matching Using the EM Algorithm and Singular Value Decomposition". *Transactions on Pattern Analysis and Machine Intelligence*, vol. 23 no. 10, IEEE, 2001, pp. 1120-1136.

[17] M. Ehrig, and Y. Sure, "Ontology Mapping - An Integrated Approach, " in *3rd The Semantic Web: Research and Applications*, LNCS 3053, Springer Berlin / Heidelberg, 2004, pp. 76-91.

[18] M. Ehrig, and S. Staab, "QOM - Quick Ontology Mapping," in *Proc. 4th International Semantic Web Conference* (ISWC05), Hiroshima, Japan, 2004, pp. 683-697.

[19] N.F. Noy, and M.A. Musen, "The prompt suite: Interactive tools for ontology merging and mapping". *International Journal of Human-Computer Studies*, vol. 59, no. 6, Academic Press, Dec. 2004.

[20] C. Fellbaum, (1998, ed.). WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

[21] A. Ghazvinian, N.F. Noy, C. Jonquet, N. Shah, and M.A. Musen, "What Four Million Mappings Can Tell You about Two Hundred Ontologies," in *Proc. 8th International Semantic Web Conference* (ISWC '09) Springer-Verlang Berlin / Heidelberg, 2009, pp. 229-242.

[22] M. Ehrig, S. Staab, and Y. Sure, "Bootstrapping Ontology Alignment Methods with APFEL," in *Proc. 4th International Semantic Web Conference* (ISWC05), Springer Berlin / Heidelberg, 2005, pp. 186-200.

[23] L. Xu, and D.W. Embley, "Discovering Direct and Indirect Matches for Schema Elements, " in *Proc. 8th International Conference on Database Systems and Advanced Applications* (DASFAA'03), IEEE Computer Society, Washington DC, USA, 2003, pp. 39-46.

[24] D.W. Embley, D. Jackman, and L. Xu, "Multifaceted Exploitation of Metadata for Attribute Match Discovery," in *Proc. Information Integration In Proceedings of the International Workshop on Information Integration on the Web* (WIIW'01), 2001, pp. 110-117.

[25] I. Witten, and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd ed., Morgan Kaufmann: San Francisco, June 2005.

[26] B. Gajderowicz, and A.F. Sadeghian, A.F. Bobillo, "Ontology Granulation Through Inductive Decision Trees," in *Proc. Workshop on Uncertainty Reasoning for the Semantic Web,* (URSW'09), Spinger-Verlang, 2009, pp. 39-50.

[27] I. Horrocks, P.F. Patel-Schneider, and F.V. Harmelen, "From SHIQ and RDF to OWL: The Making of a Web Ontology Language". *Journal of Web Semantics*, vol. 1, no. 1, Elsevier, 2003, pp. 7-26.

[28] U. Straccia, "A Fuzzy Description Logic for the Semantic Web". *Fuzzy Logic And The Semantic Web, Capturing Intelligence*, Elsevier Amsterdam, 2006, pp. 167-181.

[29] U. Straccia, "A Fuzzy Description Logic", in *Proc. 15th National Conference on Artificial Intelligence*, AAAI, Madison, USA, 1998, pp. 594-599.

[30] U. Straccia, "Reasoning within Fuzzy Description Logics". *Journal of Artificial Intelligence Research*, vol. 14, 2001, pp. 137-166.

[31] Z. Ding, Y. Peng, and R. Pan, "BayesOWL: Uncertainty Modeling in Semantic Web Ontologies," in *Soft Computing in Ontologies and Semantic Web*, Springer-Verlang, vol. 204, 2005, pp. 3-29.

[32] I. Horrocks, and U. Sattler, "Ontology Reasoning in the SHOQ(D) Description Logic," in *Proc. 17th Int. Joint Conf. on Artificial Intelligence* (IJCAI 2001), Morgan Kaufmann, 1998, pp. 199-204.

[33] J.Z. Pan, and I. Horrocks, "Reasoning in the SHOQ(Dn) Description Logic," in *Proc 2002 Int. Workshop on Description Logics* (DL-2002), Apr. 2002.

[34] K.B. Laskey, "MEBN: A Language for First-Order Bayesian Knowledge Bases, " *Artificial Intelligence*, vol. 172 no. 2-3, 2008, pp. 251-178.

[35] P. Klinov, and L.J. Mazlack, "Granulating Semantic Web Ontologies", in *Proc IEEE International Conference on Granular Computing*, IEEE , 2006, pp. 431-434.

[36] Z. Ding, Y. Peng, R. Pan, and Y. Yu, "A Bayesian Methodology towards Automatic Ontology Mapping," in *Proc. American Association Artificial International '05 C&O Workshop on Contexts and Ontologies: Theory, Practice and Applications*, AAAI, 2005, pp. 72-79.

[37] M. Leida, P. Ceravolo, E. Damiani, Z. Cui, and A. Gusmini, "Semantics-aware matching strategy (SAMS) for the Ontology meDiated Data Integration". *International Journal of Knowledge Engineering and Soft Data Paradigms* (IJKESDP), vol. 2, no. 1, Inderscience Geneva, 2010, pp. 33-56.

[38] S. Volz, "Data-Driven Matching of Geospatial Schemas," in *Spatial Information Theory*, Springer Berlin / Heidelberg, LNCS 3693, 2005, pp. 115-132.

[39] M. Ehrig, and E.B. Jérome, E.B, "Relaxed Precision and Recall for Ontology Matching," in *Proc. Workshop on Integrating Ontologies*, Banff, Alberta, Canada, 2005, pp. 25-32.

[40] B. Motik, R. Shearer, and I. Horrocks, "Hypertableau Reasoning for Description Logics". *Journal of Artifical Intelligence Research* (JAIR), AI Access Foundation, 2005, vol. 36, no. 1, pp. 165-228.

[41] M. Horridge, and S.R. Bechhofer, "The OWL API: A Java API for Working with OWL 2 Ontologies," in *Proc 6th OWL Experienced and Directions Workshop* (OWLED-08), CEUR-WS.org , vol. 529, 2008.

[42] S. Staab and R. Studer, eds., *Handbook on Ontologies*, Springer-Verlang, 2007.

[43] A. Gangemi, N. Guarino, C. Masolo, and A. Oltramari, "Sweetening WordNet with Dolce". *AI Magazine*, AAAI, vol. 24, no. 3, 2003, pp. 13-24.

[44] C. Matuszek, J. Cabral, M. Witbrock, and J. DeOliveira, "An Introduction to the Syntax and Content of Cyc", *2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, AAAI, 2006, pp. 44-49.

[45] I. Niles, and A. Pease, "Towards a Standard Upper Ontology," in *Proc. International Conference on Formal Ontology in Information Systems*, ACM, Ogunquit Maine, New York, NY, USA, 2001, pp. 2-9.

[46] R.G. Raskin, M.J. Pan, "Knowledge representation in the semantic web for Earth and environmental terminology (SWEET), " *Computer and* Geoscience, Elsevier, vol. 31, no. 9, 2004, pp. 1119-1125.

[47] M. Enkhsaikhan, W. Wong, W Liu, and M.P. Reynolds, "Measuring Data-Driven Ontology Changes using Text Mining," in *Proc. 6th Australasian Conference on Data Mining and Analytics*, Australian Computer Society, vol. 70, 2007, pp. 39-46.

[48] G. Stumme, and A. Maedche, "FCA-MERGE: Bottom-Up Merging of Ontologies," in *Proc. 7th International Conference on Artificial Intelligence* (IJCAI '01), Morgan Kaufmann, vol. 1, 2001, pp. 225-230.

[49] S.M. Falconer, N.F. Noy, and M.A. Storey, "Towards understanding the needs of cognitive support for ontology mapping, " in *Proc 1st International Workshop on Ontology Matching* (OM-2006), CEUR-WS.org, vol. 225, 2006.

[50]N. N. Guarino, and C. Welty, "Evaluating ontological decisions with OntoClean." in *Communications of the ACM*, ACM, vol. 45, no. 2, 2002, pp. 61-65.

[51] F. Giunchiglia, P. Shvaiko, and M. Yatskevich, "S-Match: an Algorithm and an Implementation of Semantic Matching," in *Proc. European Semantic Web Symposium* (ESWS'04), 2004, pp. 61-75.

[52] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, "Learning to Map Between Ontologies on the Semantic Web," in *Proc 11th International Conference on World Wide Web* (WWW'02), ACM, New York, NY, 2002.

[53] M. Brodaric, and M. Gahegan, "Experiments to Examine the Situated Nature of Geoscientific Concepts". *Spatial Cognition & Computation: An Interdisciplinary Journal*, vol. 7, no. 1, 2007, pp. 61-95.

[54] P. Langley, Editorial: *On Machine Learning. Machine Learning*, Kluwer Academic Publisher, vol. 1, no. 1, 1986, pp. 5-10.

[55] L. Breiman, "Bagging Predictors". *Machine Learning*, Kluwer Academic Publisher, vol. 24, no. 2, 1996, pp. 123-140.

[56] H. Perez-Urbina, I. Horrocks, and B. Motik, "Efficient Query Answering for OWL 2," in *Proc. 8th International Semantic Web Conference* (ISWC 2009), Springer-Verlang Berlin / Heidelberg, LNCS 5823, 2009, pp. 489-504.

[57] R.B.H. Kwok, "Translations of Ripple Down Rules into Logic Formalisms," in *Proc. 12th European Workshop on Knowledge Acquisition, Modeling and* Management (EKAW '00), Springer-Verlang London, UK, 2000, pp. 366-379.

[58] C. Erdur, and I. Seylan, "The design of a semantic web compatible content language for agent communication". *Expert Systems*, vol. 25, no 3, July 2008, pp. 268-294.

[59] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider (eds.), *The Description Logic Handbook: Theory, Implementation and Applications*, Cambridge University Press, 2003.

[60] I. Horrocks, O. Kutz, and U. Sattler, "The Even More Irresistible SROIQ," in *Proc 10th International Conference on Principles of Knowledge Representation and Reasoning* (KR2006), AAAI, 2006, pp. 57-67.

[61] L. Stojanovic, "Methods and Tools for Ontology Evolution." Ph.D. Thesis, University of Karlsruhe, Germany, 2004.

[62] M. Krotzsch, S. Rudolph, and P. Hitzler, "Description Logic Rules," in *Proc. 18th European Conference on Artificial Intelligence* (ECAI 2008), IOS Amsterdam, 2008, pp. 80-84.

[63] R. Ichise, "Machine Learning Approach for Ontology Mapping Using Multiple Concept Similarity Measures," in *Proc. 7th International Conference on Computer and Information Science* (ICIC 2008), IEEE, 2008, pp. 340-346.

[64] J. Zhang, A. Silvescu, and V. Honavar, "Ontology-Driven Induction of Decision Trees at Multiple Levels of Abstraction," in *Abstraction, Reformation, and Approximation*, Springer-Verlang, LNCS 2371, 2002, pp. 316--323.

[65] A. Bouza, G. Reif, A. Bernstein, and H. Gall, "SemTree: Ontology-Based Decision Tree Algorithm for Recommender Systems," in *Proc. 7th International Semantic Web Conference* (ISWC2008), Germany, 2008.

[66] B. Kieler, "Semantic Data Integration Across Different Scales: Automatic Learning Generalization Rules", in *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, 2008.

[67] B. Kieler, W. Huang, J.H. Haunert, and J.M. Jiang, "Matching River Datasets of Different Scales," in *Advances in GIScience*, Springer, 2009, pp. 135-154.

[68] M. Uschold, and M. Gruninger, "Ontologies and semantics for seamless connectivity," in *ACM SIGMOD Record*, ACM, vol. 33, no. 4, Dec. 2004, pp. 58-64.

[69] N.F. Noy, "Semantic integration: a survey of ontology-based approaches," in ACM *SIGMOD Record*, ACM, vol. 33, no. 4, pp. 65-70.

[70] W.S. Li, and C. Clifton, "SEMINT: a tool for identifying attribute correspondences in heterogeneous databases using neural networks". in *Data and Knowledge Engineering*, Elsevier, vol. 33, no. 1, pp. 49-84.

[71] J. Evermann, "An Exploratory Study of Database Integration Processes". in *IEEE Transactions on Knowledge and Data Engineering*, IEEE, vol. 20, pp. 99-115.

[72] P. Shvaiko, J. Euzenat, H. Stuckenschmidt, H., M. Mochol, F. Giunchiglia, M. Yatskevich, P. Avesani, W.R. van Hage, O. Svap, and V. Svátek, *Description of alignment evaluation and benchmarking results*, Technical Report, KnowledgeWeb, 2007.

[73] A. Elkiss, "An ontology for use in converting METAR and TAF reports to DAML," Retrieved August 10, 2010 from http://www.csd.abdn.ac.uk/~ggrimnes/AgentCities/WeatherAgent/weather-ont.daml.

[74] National Weather Service Internet Web Team, "METAR Data Access," Retrieved August 10, 2010 from http://weather.noaa.gov/weather/metar.shtml.

[75] Internet Services Group, "Meteorological Character Codes," Retrieved August 10, 2010 from http://www.nws.noaa.gov/tg/code.html.

[76] D. Connolly, F. van Harmelen, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L.A. Stein, "DAML+OIL (March 2001) Reference Description," Retrieved August 10, 2010 from http://www.w3.org/TR/daml+oil-reference.

[77] C.J. van Rijsbergen, "In Information Retrieval," Butterworth, 1979.

[78] I. Horrocks, B. Parsia, and U. Sattler, "OWL 2 Web Ontology Language Direct Semantics," Retrieved on August 10, 2010 from http://www.w3.org/TR/owl2-direct-semantics.

## Glossary

There are several common terms and symbols, which have varying definitions in the literature, utilized throughout this document. To avoid ambiguous interpretations and repetitive reclassification of these terms and symbols, the following lists provide meanings and notations of the terms and symbols adopted in this document. Several acronyms and variable naming conventions have been adopted in this thesis. These are listed in the *Acronym and Conventions* section below.

## Terms

**anchor:** An *anchor* is a concept in the local ontology which was identified as a match with a target concept. Used to align ontologies and possibly perform further matches using structural or any other means.

**attribute**: It is common in database literature to refer to database fields as *columns*, *attributes*, or *properties*. This thesis uses the term *attribute*, to distinguish it from OWL properties.

**bagging:** *Bootstrap aggregation*, also known as *bagging*, is a method for improving the prediction model over some data set. Bagging works by creating several models, and assigning weights to class values that are biased against bad performing classifications, in each model.

**bagging with costs:** *Bagging with costs* is a type of *bagging* which considers probabilities of each prediction model and averages those probabilities to create an overall better model.

**boosting:** *Boosting* is a technique that "boosts" the importance of individual records in a dataset by assigning weights to those records based on whether it was correctly classified by the derived decision tree.

**class**: A *class* is classification value used by a machine learning algorithm.

**cluster:** A *cluster* is the total area on a *n*-dimensional plane covered by a particular concept's data points for *n* attributes of that concept. Each cluster is broken up into sub-clusters called *granules*, which define their own area in terms of *rules* defining a range, and *data-points* that exist within that range.

**concept**: A *concept* represents a particular element in an ontology. A concept hierarchy may be *Rain* $\sqsubseteq$ *Precipitation* $\sqsubseteq$ *WeatherEvent*. Often this is referred to as a *class*. *Concept* is adopted here to differentiate it from the data-mining term *class*, which refers to classification, as noted above.

**consistency:** *Consistency* can refer to logical consistency, where logical theory must have no contradictory axioms, or the consistency between patterns displayed amongst the records of complementary datasets. The context in which the term is used will make the distinction apparent.

**effort:** *Effort* represents the number of edges between two concepts in a single branch (**Definition 22**).

**granule**: A *granule* represents a leaf node in a decision three. See **Definition 3** and **Definition 4** for a detailed definition of a *granule*.

**ground term:** A *ground term* is a concept at the lowest level of the ontology, with no sub-classes.

**inverse role:** An *inverse role* is a role which is the reverse of another role; e.g. *isChild* is the inverse of *isParent*.

**key terms:** *Key terms* are terms that have been associated with a set of records in a database.

**option tree:** An *option tree* is a type of decision tree that, in addition to the standard decision node which leads to a single branch, also has an option node. The option node leads to all branches simultaneously, with each lead having a positive (correct) and negative (incorrect) score. The best (most correct) path is chosen as the predicate for that spot in the tree.

**qualified number:** A *qualified number* is a number which restricts the range of a role for a particular concept such as *Female*, as in the following example; e.g. Having at least 2 female children ( $\leq 2$ *hasChild.Female*).

**local**: When referring to matching any parts of a system, the primary system is referred to as the *local* system, and identified by the letter *L*.

**OWL:** OWL is the Web Ontology Language, adopted by the World Wide Web consortium (W3C). OWL 2 is the latest version of OWL.

**stacking:** *Stacking* is a technique that compares models built with different classification algorithms.

**target**: The system which the *local* system is being matched to is referred to as the *target* system, and identified by the letter *T*.

**RDFS:** *Resource Description Framework Schema* is a general-purpose language, in XML syntax, for representing assertions (or more generally information) on the Web, as proposed by W3C[48].

**Ripple Down Rule:** RDR, or *Ripple Down Rule* is an automated technique for dealing with a growing rule-based knowledge base. RDR updates an existing knowledge base by refining existing rules with more refined extensions. For example, a rule such as "a square has four sides" could be refined by attaching the rule "all sides are equal in length", giving the more accurate definition "a square has four sides, and all sides are equal in length".

**unqualified number:** An *unqualified number* is a number which restricts the range of a role, without the need to specify a particular concept; e.g. Having at least 2 children ( $\leq 2$ *hasChild*).

**upper ontology:** An *upper ontology* is an ontology that describes general concepts, and often includes common sense definitions. It is meant to support broad semantic interoperability between various ontologies.

[48] http://www.w3.org/TR/rdf-schema/

## Symbols

| | | | |
|---|---|---|---|
| ::= | definition | A ::= *definition* | A is defined by the phrase or equation "*definition"*. |
| $\sqsubseteq$ | subsumption | A $\sqsubseteq$ B | Concept A is subsumed by concept B. |
| $\sqsubseteq_{direct}$ | direct subsumption | A $\sqsubseteq_{direct}$ B | Concept A is a directly subsumed by concept B. |
| $\sqsupseteq$ | reverse subsumption | A $\sqsupseteq$ B | Concept A subsumes concept B. |
| $\sqsupseteq_{direct}$ | direct reverse subsumption | A $\sqsupseteq_{direct}$ B | Concept A directly subsumes concept B. |
| {...} | set elements | A = {*a*, *b*, *c*} | A is a set whose elements are *a*, *b*, and *c*. |
| $\{a_k\}$ | set elements over *k* | A = $\{a_k\}$ | A is a set or subset whose elements are uniquely indexed using *k*, such $a_0$, $a_1$, … |
| ω | generalization method | ωP(a, b) or $P_\omega$ | ωP or $P_\omega$ are generalized version of P. |
| ⇑ | graph siblings | A ⇑ B | Concepts A and B are subsumed by the same direct super-class. |
| $\doteq$ | concept definition | A $\doteq$ B | Concept A is defined to be equal to Concept B. |
| ≡ | concept equivalence | A ≡ B | Concept A is equivalent to concept B. |
| $\equiv_{dp}$ | data attribute equivalence | *a* $\equiv_{dp}$ *b* | Data attribute *a* is equivalent to data attribute *b*. |
| $\Rightarrow_G$ | granule implication | *Ag* $\Rightarrow_G$ *AC* | *Ag* is a granule that represents a leaf node that classifies concept *AC* in a decision trees. |
| ∧ | logical AND | A ∧ B | True if A and B are true. |
| ∨ | logical OR | A ∨ B | True if either A or B are true. |
| ∩ | set intersection | A ∩ B | Elements which exist in both set A and set B. |
| ⊓ | concept intersection | A ⊓ B | Concept that is of type A and B. |

| | | | |
|---|---|---|---|
| ¬ | logical not | ¬ A | If A is true, then ¬ A is false. |
| ∈ | set element | $m \in$ M | *m* is an element of set M. |
| ∑ | summation | ∑ *m* | Sum of elements m, in some predefined set. |
| [ 0, 1 ] | normalized range | [ 0, 1 ] | Continuous *real* numeric values in the range 0 to 1 inclusively. |

## Acronyms and Variable Name Conventions

Any label in capital letters, such as GAP_MIN or TOP_RANK, is a user-defined *constant*, *relation* or *set*. The type of value and its usage will be apparent through its name and the context within which it is used. A user-defined variable is labelled with more than one word, and these words will be connected with an underscore "_". If they are not, this represents either a *relation* or a *set*. The naming convention for relations and sets is the acronym of the relation or the set's title. Relations are presented in an italic font. Sets of relations have the same label as the relation, are presented in non-italic font, and have the subscript "set". For example, the "Close Match" relation *CM* is a member of the set $CM_{set}$, and each set has a formal definition. Each *constant*, *relation* and *set* acronym is defined below. A *collection* is a set of sets, and is identified by the special Blackletter font such as $\mathfrak{CM}$.

**$C2CM_{set}$:** "Local Concept To Target Concept Match" is the set of all *ConceptMatch* relations (**Definition 7**).

**CLASS_NODE_MIN_FACTOR:** The minimum number of records in a decision tree leaf node. Ensures equal precision of each leaf node, and limits the number of leaf nodes created.

***CM*:** A "Close Match" relation represents a match based on a high ranking *MRank* relation (**Definition 15**).

**$CM_{set}$:** The set of all *CM* relations (**Definition 15**).

**$\mathfrak{CM}$:** The collection of all *CMSubs* relations (**Definition 16**).

**CMSubs:** The set "Close Match SubClasses" is a set containing all *CM* relations for a target concept *TC*, grouped under the most general local concept, for ease of readability during verification by the user, and is written as $CMSubs_{TC}$ (**Definition 16**).

***ConceptMatch*:** A "Concept Match" relation represents a set of concept matches based on their matched granules in $P2RM_{set}$ (**Definition 6**).

**$CP_{label}$:** "Correspondence Proximity" is a particular characteristic of MCP, as indicated by the subscript *label* (**Definition 20**).

***CVPM*:** A "Child via Parent Match" relation is a match between concepts based on overlapping granules and a "Gap To Parent Match" between their parent concepts (**Definition 14**).

**$CVPM_{set}$:** The set of all *CVPM* relations (**Definition 14**).

**GAP_CHECK_MIN:** The *MRank* threshold used to determine if a "Gap Match" relation between two concepts is an element of $GM_{set}$. Ensures two low ranking matches are not included in $GM_{set}$ match. See section 3.12 for an example.

**GAP_MIN:** The *MRank* gap threshold used to find a Gap Match (GM) between two concepts.

***G2PM*:** A "Gap To Parent Match" relation represents a match between two parent concepts based first on their children's *GM* relation, and secondly on the closeness of their own *MRank* relation (**Definition 13**).

**$G2PM_{set}$:** The set of all *G2PM* relations (**Definition 13**).

***GM*:** A "Gap Match" relation represents a match between two concepts, highly distinct from other matches, i.e. having *MRank* ranking much higher then the next match (**Definition 12**).

**$GM_{set}$:** A set of *GM* relations (**Definition 12**).

**L2TM:** "Local To Target Matches", a set of all selected proposed matches, grouped by local concepts (**Definition 17**).

***LC* vs. *TC***: In general, local and target ontology concepts will be represented by *LC* and *TC*, respectively. For simplicity, on some occasions concepts may be referenced by their identifier, meaning $LC_X$ may be represented by simply *X*, especially in figures. When mentioning a term used by both *local* and *target* systems, a unified scheme is used to define *AC*, where $A \in \{L, T\}$. The letter will be defined before its use.

***Lg* vs. *Tg*:** A local and target granule (**Definition 3** and **Definition 4**).

**MCP:** "Match Correspondence Proximity" is a proximity measure between two matches. Used for algorithm evaluation purposes (**Definition 20**).

**$MCP_{full}$:** "Match Correspondence Proximity - Full" is the aggregation of individual MCP measures for each match (**Definition 21**). This is the final rating used to evaluate a particular configuration of proximity measures.

***LM*:** A "Level Match" relation between a local and a target concept at different levels of the local ontology (**Definition 10**).

**$LM_{set}$:** The set of all *LM* relations between local concepts and a single target concepts *TC*, for some local ontology level $\lambda$, written as $LM_{TC\lambda}$.

**MS:** "Match Score" is a function giving the quality of matches between a local and target concept (**Definition 9**).

***MRank*:** A "Match Rank" relation associates a local and target concept and *LM* relation (if one exists) with a numeric rank, that is a normalized value in the range [ 0, 1 ] , for some local ontology level $\lambda$ (**Definition 11**).

**$MRank_{set}$:** The set of all *MRank* relations (**Definition 11**).

***P2RM***: A "Points To Rule Match" relation represents a match between the rule and data-points of two overlapping granules (**Definition 5**).

**$P2RM_{set}$:** When referring to matched concepts *LC* and *TC*, $P2RM_{set}$ is the set of all *P2RM* relations between granules that imply either *LC* or *TC* (**Definition 6**).

**PROP_CLUSTER_RANGE:** A factor used to expand granule *rules* and decrease the coverage of *data-points*.

**T2LM:** "Target To Local Matches" is a set of all selected proposed matches, grouped by target concepts (**Definition 17**).

**TOP_RANK_MIN:** The value of a threshold that determines a "Close Match" (*CM*) between two concepts, based on the rank associated with the *MRank* relation between those concepts.

**TOP_P2P_RANK_MIN:** The value of a threshold that determines if a Gap To Parent Match (*G2PM*) relation is found between two concepts, based on the rank associated with the *MRank* relation between those concepts.