2012

Towards High Quality Semantic Web Data: Detecting Abnormal Data on the Semantic Web

Yang Yu
Lehigh University

Recommended Citation

Yu, Yang, "Towards High Quality Semantic Web Data: Detecting Abnormal Data on the Semantic Web" (2012). *Theses and Dissertations.* Paper 1373.

TOWARDS HIGH QUALITY SEMANTIC WEB DATA: DETECTING ABNORMAL DATA ON THE SEMANTIC WEB

by

Yang Yu

A Dissertation
Presented to the Graduate Committee
of Lehigh University
in Candidacy for the Degree of
Doctor of Philosophy
in
Computer Science

Lehigh University
May 2012

This dissertation is accepted in partial
fulfillment of the requirements for the degree of
Doctor of Philosophy.

(Date)

Jeff Heflin (Chair)

Brian Davison

Donald Hillman

Lin Lin

Acknowledgements

I would like to thank Professor Jeff Heflin for all his help. I cannot thank him enough for his support and patience. He is a great advisor and I have been inspired by his brilliance, hard work, and dedication and have benefited greatly from his guidance and example. Without him this work would not have been possible. I would like also to thank my committee members, Professors Brian Davison, Donald Hillman and Lin Lin for their thoughtful comments and encouragement. I am also warmly grateful to my colleagues Yingjie Li, Xingjian Zhang, Dezhao Song and Zhengxiang Pan for insightful suggestions and helps. Finally this dissertation is dedicated to my wife Yun Sun, my parents Yukun Yu and Yafen Huang and my parents-in-law Changqing Sun and Xianzhi Zhang. Without their endless love, patience, understanding and support I would certainly not be where I am today.

Contents

List of Tables

List of Figures

Abstract

In an information system spanning multiple, distributed, and autonomous data sources, data quality is a problem intrinsic to any architecture of an integrated information system, because the providers of the data have control over their source content and how they describe it. Low quality data is also a pressing problem for consumers of distributed information. Because recent developments in the Semantic Web have suggested that it may be possible to rethink information integration, data quality research on the Semantic Web can be promising to solve the quality issues in distributed information systems. Correctness is often used synonymously with data quality.

The goal of this work is to design algorithms to detect erroneous Semantic Web data by identifying abnormality, because such abnormal data is indicative quality issues. Such algorithms would be very useful in many scenarios, e.g., filtering query results derived from low quality data, providing input for other assessments (e.g. trust) and improving the quality of the data in integrated systems. One means of assessing quality is finding corroborations, e.g. an axiom that can be entailed by other axioms is more likely correct, because the entailment can be seen as a corroboration. Similarly, we have the probabilistic rules that are valid for most or verified data and a statement is entailed by these rules, then that statement is more likely correct than those that cannot be entailed.

Based on these ideas, I developed the following algorithms. Utilizing a referenced data set that is assumed to contain few errors and where the closed world assumption is valid, the first algorithm tries to learn to classify data into categories for each type of error that an object property triple could have. The second algorithm

focuses on relaxing the closed world assumption, i.e. the statements not existing in data can not be assumed false. Without learning from a referenced data set in advance, the third algorithm discovers the patterns that are similar to the ones used in the previous systems, but relaxes the assumption of ”few errors”. Then it improves on three aspects compared to the previous systems. (1) The process of searching patterns is more efficient than the previous systems by doing a level wise searching; (2) the probabilities of patterns are affected by the data with different truth probabilities; (3) the system checks logic consistencies among patterns to further differentiate them. The last algorithm tries to discover value-clustered graph functional dependencies, an extended concept of functional dependency in databases. These dependency rules have a more general form than the patterns in the other systems, and can capture more latent semantics in data. Using them, the system greatly improves the capability of detecting abnormality on all types of values and in the situation where no explicit connections exist in data. Experiments on a number of data sets from different domains validate these systems. All these algorithms can be easily applied to common Semantic Web data in query answering systems, information integration systems and semantic search systems.

Chapter 1

Introduction

It is clear that we are living in an age of increasingly vast digital information. Quite literally, data is everywhere. Meanwhile, of course, this proliferation of data is not expected to stop anytime soon. However data is deemed of only high quality if they correctly represent the real-world construct to which they refer. Although there are many definitions of Data Quality (DQ), the following is often used:

> ...consistently meeting all knowledge worker and end-customer expectations in all quality characteristics of the information products and services required to accomplish the enterprise mission (internal knowledge worker) or personal objectives (end customer) [28].

The consequences of poor quality of data are often experienced in everyday life, but, often, without making the necessary connections to their causes. For example, the late or mistaken delivery of a letter is often blamed on a problematic postal service, although a closer look often reveals data-related causes, typically an error in the address, originating in the address database. A more significant error is related to the potential health impacts of radiation exposure that are often a source of concern for people. It is reported in Wired.com [57] and CNN [4] that the Transportation Security Administration (TSA) in the United States is to conduct extensive radiation safety tests on their introduced backscatter full body scanners (affectionately known as the “nudie scanner” in some quarters). An internal review of the previous safety testing which had been done on the devices revealed a litany of *calculation errors*, *missing data* and *other discrepancies on paperwork*. In short, it is a data quality problem. A TSA spokesperson described the issues to CNN as being "record keeping errors". The errors affected approximately 25% of the scanners which are in operation, which Wired.com identifies as being from the same manufacturer, and included errors in the calculation of radiation exposure that occurs when passing through the machine. The calculations were off by a factor of 10. Wired.com interviewed a TSA spokesperson and they provided the following information: "Rapiscan technicians in the field are required to test radiation levels 10 times in a row, and divide by 10 to produce an average radiation measurement. Often, the testers failed to divide results by 10." For their part, the manufacturer is redesigning the form used by technicians conducting tests to avoid the error in the future. There are more examples showing that the quality of data has serious consequences, of far-reaching significance, for the efficiency and effectiveness of organizations and businesses. As the report on data quality of the Data Warehousing Institute (see [27]) estimates that data quality problems cost U.S. businesses more than 600 billion dollars a year, because there is a significant gap between perception and reality regarding the quality of data in many organizations.

With the information explosion, information integration - the merging of information from disparate sources with differing conceptual, contextual and typographical representations - is a natural and necessary requirement. However data quality becomes more critical and harder to solve in the context of information integration. As in an information system spanning multiple, distributed, and autonomous data sources, data quality can suffer problems that are intrinsic to any architecture of an integrated information system as well as various technical problems that arise from an integrated system [73]. Data sources are autonomous from the integrators point of view. Due to autonomy, source accessibility varies. Some sources allow full query capability, some provide only simple protocol connect (e.g. html-forms). Thus, not all available data can actually be accessed and used. Also due to source autonomy, sources tend to be heterogeneous in various aspects: Sources use different data models, have different semantics, such as attribute names, different scope, different structures, etc. This heterogeneity decreases the quality of an integrated result.

To support information integration, a significant portion of traditional information technology expenditures is used to translate information from one format to another, thus enabling exchange of information between units and systems. Recent developments in the Semantic Web have suggested that it may be possible to rethink information integration - to integrate sources on-the-fly, as opposed to developing special purpose translation programs. Semantic Web technologies aim to attach data structure, typed links, and axiomatically represented implicit facts to data available on the Web. One of the goals is to empower computers to better extract, combine, interpret, and reuse the data [16]. Linked Data is a representative Semantic Web data cloud (shown in Figure 1.1). It currently consists of 31 billion RDF triples, which are interlinked by around 504 million RDF links (September 2011), and is growing exceptionally fast [17]. A major share of such data originates from existing relational databases and is lifted by mapping database schema elements to Web ontologies. An ontology is a formal logic based description of a vocabulary that allows one to talk about a domain of discourse. The vocabulary is articulated using definitions and relationships among the defined concepts (I will introduce more about ontology and related knowledge in Chapter 2). Businesses and public institutions have already started to publish significant amounts of real world data on the Web using Web ontologies [36]. In addition to the growing number of data published directly by the owners of the data source, there is development of tools that actively retrieve real data from vendors and provide a Semantic Web interface. For example, the enterprise OpenLink Software has released a middleware technology called "Sponger cartridges" that creates, on the fly, RDF representations of Amazon, eBay, and other commerce sites using the GoodRelations ontology [47] by accessing vendor-specific APIs [1]. This makes more unprecedented amounts of actual business data available on the Web of Linked Data. Furthermore many real world applications are beginning to use Semantic Web techniques and exploit such Semantic Web data. Here are some examples of them.

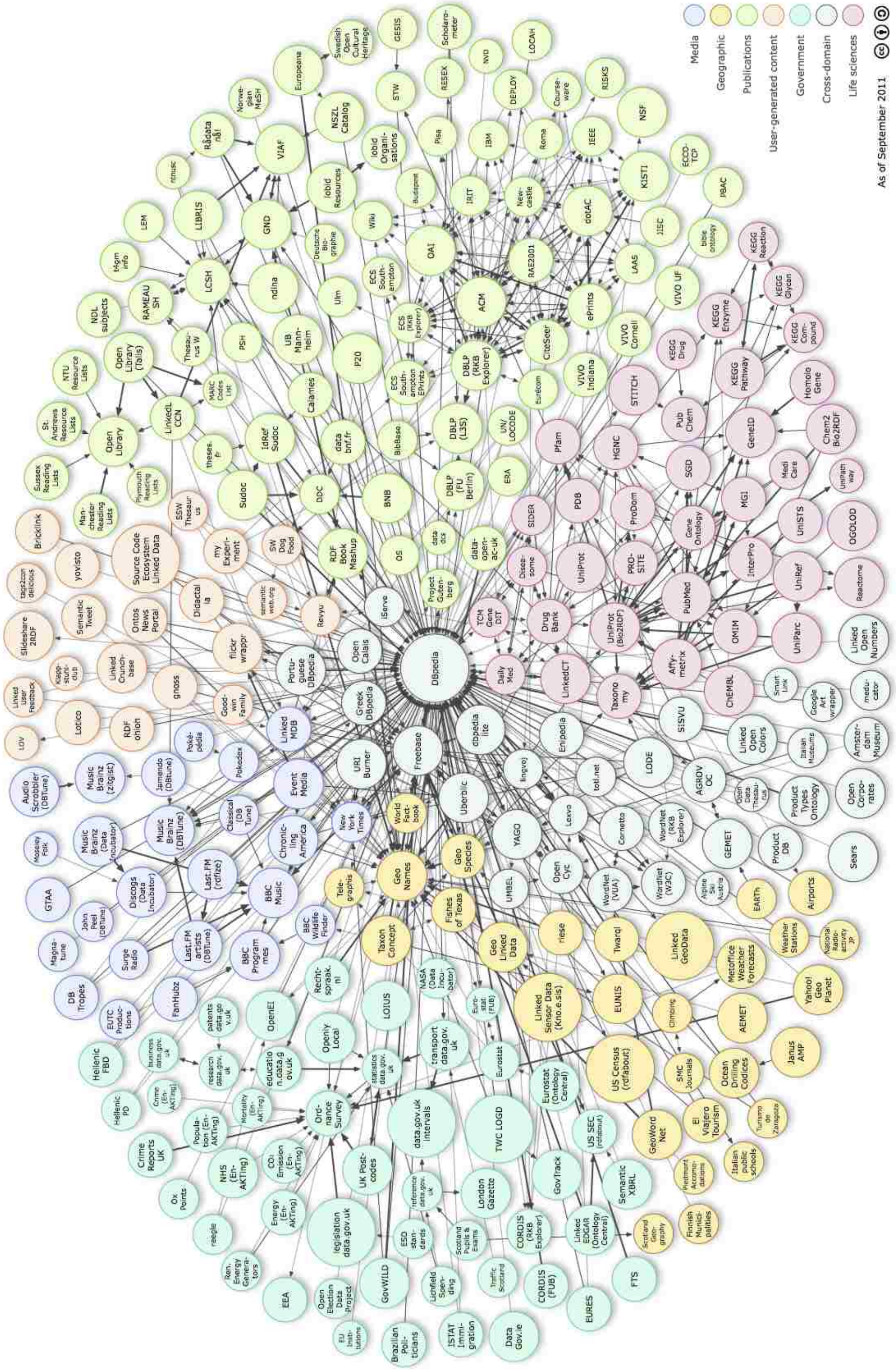


Figure 1.1: Instance linkages within the linking open data datasets

- Life sciences domain. As many biological datasets are presently available in tabular format, a prototype web-based application called YeastHub [23] demonstrates how a life sciences data warehouse can be built using a native Semantic Web data store. This data warehouse allows integration of different types of yeast genome data provided by different resources in different formats including the tabular and RDF formats. Once the data is loaded into the data warehouse, queries can be formulated to retrieve and query the data in an integrated fashion.

- E-commerce domain. Internet business-to-business transactions present great challenges in merging information from different sources. Yu et al. [102] describe a project to integrate four representative commercial classification systems with the Federal Cataloging System (FCS). The FCS is used by the US Defense Logistics Agency to name, describe and classify all items under inventory control by the Department of Defense. Our approach uses the ECCMA Open Technical Dictionary (eOTD) as a common vocabulary to accommodate all different classifications. Then we create a semantic bridging ontology between each classification and the eOTD to describe their logical relationships in a semantic web language. The essential idea is that since each classification has formal definitions in a common vocabulary, subsumption can be used to automatically integrate them, thus mitigating the need for pairwise mappings. Furthermore the system provides an interactive interface to let users choose and browse the results and more importantly it can translate catalogs that commit to these classifications using compiled mapping results.

- Social media domain. Semantic Wikis have demonstrated the power of combining Wikis with Semantic Web technology. The KiWi system goes beyond Semantic Wikis [86] by providing a flexible and adaptable platform for building different kinds of Social Semantic Software, powered by Semantic Web technology. While the KiWi project itself is primarily focused on the knowledge management domain, this demonstration shows how KiWi aspects like the Wiki Principles and Content Versatility can be used to build completely different kinds of Social Software applications. The first application shown in this project is an "ordinary" Semantic Wiki system preloaded with content from a online news site. The second application called TagIT is a map-based system where locations and routes on a map can be "tagged" by users with textual descriptions, SKOS categories, and multimedia material. Both applications are built on top of the same KiWi platform and actually share the same content.

With a growing amount of Semantic Web data, as observed by more and more consumers of such data, there are numerous quality problems in Semantic Web data. For example, DBpedia is a project aiming to extract structured content from the information created as part of the Wikipedia project. DBpedia has been described by Tim Berners-Lee, inventor of the WWW (including URIs, HTTP, and HTML), as one of the more famous parts of the Linked Data project [15], as DBpedia allows users to query relationships and properties associated with Wikipedia resources, including links to other related datasets. Many quality issues are found in DBpedia data set, e.g.

1. <http://www.dbpedia.org/resource/Harrow_College,
   http://www.dbpedia.org/ontology/School/upperAge,
   2009.0>,

2. <http://www.dbpedia.org/resource/Wake_Island,
   http://www.dbpedia.org/ontology/Island/country,
   http://www.dbpedia.org/resource/United_States_Air_Force>.

The first piece of RDF data listed above means the maximum age for being enrolled in Harrow College is 2009. The second means the country where Wake Island is located in United States Air Force. The consequence for the Semantic Web will be costly, if we have low quality data, because wrong answers to queries might make intelligent agents making incorrect decisions on behalf of their users or directly make human who believe these answers to take wrong actions. Therefore data quality research on the Semantic Web becomes critically important for development of both the Semantic Web itself, information integration and further the whole Web of information. Due to the problem of scale, e.g. DBpedia has over 10 million RDF triples, it is impractical to have a human “scrub” all of the data. An automated approach is needed, although few research efforts yet have been devoted in this area.

1.1 The Semantic Web

There are two key ideas to the Semantic Web: a flexible model for representing information and a formal method of expressing the meaning of a vocabulary. The first idea is expressed by the Resource Description Framework (RDF), a World Wide Web Consortium (W3C) recommendation. Essentially, RDF is a directed, labeled graph similar to a semantic network. Its main distinguishing features are an eXtensible Markup Language (XML) serialization syntax and the use of Uniform Resource Identifiers (URIs) to name things. Often, URIs are simply Uniform Resource Locators (URLs), which can make it possible to retrieve more information about the resource by retrieving information from the Web. It should be noted that RDF could be used to express the information contained in a database or spreadsheet, but is also flexible enough to express information with less regular structure. The concept of an RDF property is a relation between subject resources and object resources.

The second idea of the Semantic Web - formally defining a vocabulary - is provided by ontologies. An ontology is ”a logical theory that accounts for the intended meaning of a formal vocabulary” [41]. The Web Ontology Language (OWL) is based on Description Logics (DL) and is compatible with RDF. In OWL, it is possible to define a US computer as a computer who has at least something made in US (written USComputer $\equiv$ Computer $\sqcap \exists$ madeIn.US in standard description logic notation). Note, the Semantic Web vision does not presuppose that there is a single shared ontology; on the contrary it assumes that there are many ontologies that are interlinked. These ontologies may reuse terms from other ontologies, define new terms using terms from other ontologies or simply express relationships between their terms and those in other ontologies. Nevertheless, choosing to use ontologies does not automatically solve the information integration problem. As a matter of fact, there are now tens of thousands of heterogeneous public ontologies. However, these ontologies can be aligned using the same kinds of axioms that are used to define the semantics of terms within ontologies. The alignments may be provided by the original ontology author herself, or may be published by a third party who saw a need to integrate ontologies.

The Semantic Web is a mesh of information linked up in such a way as to be easily processable by machines, on a global scale. You can think of it as being an efficient way of representing data on the World Wide Web, or as a globally linked database. The semantic web data model has some commonalities with the model of relational databases. A relational database consists of tables, which consists of rows, or records. Each record consists of a set of fields. The record is nothing but the content of its fields, just as an RDF node is nothing but the connections: the property values. The mapping is very direct:

- a record is an RDF node;
- each field (column) name is an RDF property;
- the record field (table cell) is a value.

But RDF is much more flexible than relational tables. First, the current database systems do not permit a large numbers of columns in a table. For example, DB2 and Oracle have a limit of 1012 columns. Using the RDF model, we can have an arbitrary number of properties. Second, relational tables often have nulls in many fields. Because the relational model states that every attribute has a value for a given occurence (row/tuple) and sometimes the value is unknown, relational tables have to put null as the value in the column. In addition to creating storage overhead, nulls increase the size of the index and they sort high in the database index. However the RDF model contains tuples for only those attributes that are present in an object. Third, in case of frequent altering of the table to accommodate new data and requirements, the schema evolution is expensive in the current database systems. One of reasons why effective support for schema evolution is challenging for databases is that schema changes may have to be propagated, correctly and efficiently, to instance data, views, applications and other dependent system components. Ideally, dealing with these changes should require little manual work and system unavailability. For instance, changes to a database schema S should be propagated to instances data and views defined on S with minimal human intervention. Comparatively, schema evolution in RDF model is easy by simply adding new tuples corresponding to new attributes.

Besides the above flexibilities, one of the more important driving forces for the Semantic Web, has always been the expression, on the Web, of the vast amount of relational database information in a way that can be processed by machines. Ontologies used in the Semantic Web, can be compared with database schemas. Just like the schemas, they are essentially a data definition mechanism. However, as ontologies are logic-based, they have stronger semantics than schemas and are therefore more powerful in expressing relationships between various data attributes. Since ontologies provide a shared and unambiguous understanding of the relevant domain in a structured format, it makes it easier to automatic integrate different data sources expressed using ontologies.

1.2 Quality Assessment on the Semantic Web

Corresponding to the two key ideas of the Semantic Web, the data quality assessment on Semantic Web data falls into two categories: evaluation on ontologies and on instance data. But they are not totally independent and each may help and rely on the other. For example, to evaluate instance data, how the ontology that instance data conforms to is defined will affect the design of the evaluation process. On the other hand, the patterns or rules used in instance data evaluation certainly can be valuable for modifying and enriching ontology. The quality of an ontology may be evaluated by focusing on the different levels of the ontology. On the lexical level, string matching can be used to compare concept identifiers used in ontology with a “gold standard” set of strings that are considered a good representation of concepts of the problem domain. On the structure level it often requires manual intervention by a trained human expert familiar with some philosophical notions, such as essentiality, rigidity, unity; the expert should annotate the concepts of the ontology with appropriate metadata tags, whereupon checks for certain kinds of errors can be made automatically [20]. While the above two methods both need significant human effort, the last one is to use automatic logic inference, such as consistency and satisfiability checks, provided by appropriate reasoners.

The ontology serves as the vocabulary for describing instance data. Instance data is a set of triples that is used to describe individuals. Since instance data is the dominant part of the real world knowledge, it is more important to develop a system to evaluate instance data quality and to determine when certain data is ready for use. Because instance data often accounts for orders of magnitude more data than ontology data and is more distributed and more error prone, it is impractical to measure its data quality using approaches that need too much human effort, like the first two methods for ontology evaluation described above. Thus we need to automate the DQ measurement process. However, most real world instance data does not provide a solid basis for applying the automatic approach similar to that for ontology evaluation. There could be many reasons and some of them are missing ontologies, poor ontology design, misunderstanding of ontology use, missing or incomplete instance data, etc. Thus one of main motivations of my work is to devise mechanisms to automate evaluation of Semantic Web data with minimal requirement for precise and rigid ontology use.

Data quality research is important for the Semantic Web. One of the most exciting things about the Semantic Web is the potential of moving Web search from document relevance to query answering, because the Semantic Web knits knowledge together with logic meanings instead of loosely structural linked as it in the Web. It would be possible for computer algorithms to better interpret the data and use it differently in query answering according to its quality. Thus there are several imperative applications requiring correctness evaluation on triples, for example filtering query results derived from low quality data, checking the quality of a new dataset before integration and serving as input for other assessment, like trust.

1.3 Contributions

Since the Semantic Web represents many points of view, there is no objective measure of correctness for all Semantic Web data. I designed algorithms that detect potentially erroneous data by identifying data that appears to be anomalous. When designing this kind of systems, I first determined if useful patterns do exist. Then I realized that it is required to devise mechanisms to discover these patterns in relatively easier situations, i.e. where the patterns are strong and rich in the data. After that, I gradually improved the capability of discovering more potential patterns and adapted the requirement on the data for learning in different situations. The technical problem this dissertation addresses can be summarized as follows: given a Semantic Web data set that uses terms from any ontology of the user's choice, identify aberrant data in it that could be potentially erroneous, even if I cannot learn from any prior data that commits to these ontologies. In providing a solution to this problem, I have developed a framework that allows us to investigate this problem space in a formal manner. Further I have designed, implemented and evaluated four algorithms that can solve this problem in different situations. My dissertation specifically makes the following technical contributions:

- I have developed three algorithms to evaluate the data quality issues of object property triples in different situations according to completeness and entire quality of data. The first one is based on the closed world assumption. The closed world assumption (CWA) treats the statements that are not in the data as false. This algorithm is sufficient when the data set is well described, i.e. almost all the instances in the domain have been given the value representing their real world status for every property that can be applied on them. If the data is not that well described, i.e. some facts are missing from the data set, the second algorithm is developed for the situation where the open world assumption (OWA) is applied. The open world assumption is the opposite of the closed world assumption. The absence of a particular statement within the web means, in principle, that the statement has not been made explicitly yet, irrespectively of whether it would be true or not, and irrespectively of whether we believe (or would believe) that it is (or would be) true or not. This situation is more common in real world data sets than the previous situation. The last algorithm for evaluating object property triples is mainly designed to deal with a data set that may have a significant portion of erroneous data. It needs the system to take into account more aspects, such as the prior truth probability of data from which the rules are learned and logic consistency among the rules learned. Compared among these three algorithms, the one based upon CWA has the fastest learning and minimal space requirement. The one based upon OWA performs the best in OWA situation but has the worst space cost. The strength of the third algorithm is to deal with noisy data and has an iterative process. Using a new data structure to summarize the RDF graph, it still can be done within reasonable amount of time, e.g. one hour for SWRC. Therefore each one has its special advantages and use cases.

- I have demonstrated space of connections, and dependency rules, can contribute to the evaluation of object property triples in order to increase accuracy of data quality problem. For clear demonstration, I developed different mechanisms to represent a useful context, while they are based on similar essence. In the algorithm designed for the data set under CWA, the context is constructed as an RDF graph. In the algorithm designed for data sets under OWA, the context is expanded to capture more and similar information by merging the context of similar pairs of objects in the data. In addition, the representation of context is simplified into semantic connections for the data set that can have a significant portion of erroneous data. The semantic connection is a sequence of labels on paths in an RDF graph. In the third algorithm for evaluating object property triples, I further improved the process of constructing context from per triple basis to a more efficient per graph basis. Compared between the representations using graph and using semantic connections, the former one needs much less space, since every edge is stored once. However the latter one stores the semantic connections between pairs of nodes and different semantic connections may share some of the same edges.

Thus the latter one costs more space but contributes to better accuracy of data quality problem by supporting to consider similarity between different semantic connections.

- I have extended the concept of functional dependency from relational databases into RDF graphs and used them to detect abnormal Semantic Web data. The algorithm using the extended concept of functional dependency is so general that it can deal with all data types of Semantic Web data, as opposed to only object property triples. The algorithm also finds natural clusters of values of each property, i.e. the values in a cluster have similar semantics. Further, using these value clusters, the algorithm searches for value-clustered graph functional dependency rules in Semantic Web data. Using valued-clustered graph functional dependency rules, the system can uncover significantly more erroneous data than the prior algorithms that can only be exposed after grouping values that are similar in semantics. Comparing this algorithm with the prior three algorithms coping with object property triples, each of them has its own advantages. The prior algorithms are based on context which gives more information and so can derive better recall. Recall in information retrieval is the fraction of the documents that are relevant to the query that are successfully retrieved. Here I use recall to measure the fraction of true errors that are reported as abnormal by the system. Further the other algorithms are better on detecting relational errors. The algorithm extending functional dependency has the most general capability, since it can deal with both object property triples and datatype property triples.

1.4 Thesis Overview

My detailed approach is described throughout this document, however, it maybe useful to give a high level view of my approach and some considerations behind it early on. In my research I need to answer the following questions:

1. Which different situations are there in real world Semantic Web data sets, for

example whether the descriptions are relatively complete given the domain that the data is focus on, whether the data set can be assumed generally correct, etc.

2. What information do I use to describe the context of Semantic Web data and how the context is useful for evaluating the data?

3. How do I learn the common characteristics in the contexts for similar data and how do I present the characteristics into rules to build classifiers evaluating Semantic Web data in different situations?

4. How do I construct these classifiers based upon learned rules and how these classifiers can be used for differentiating normal and abnormal data?

The answer to the first question listed above is mainly discussed in Chapter 3. Given the problem, I classify real world data sets into several situations and then I give design considerations of a practical system for each of them. From a machine learning perspective, the system essentially is an unsupervised learning system. Unsupervised learning refers to the problem of trying to find hidden structure in unlabeled data. Since the examples given to the learner are unlabeled, there is no error or reward signal to evaluate a potential solution. In machine learning theory, the characteristics of a data set significantly affect the design or choice of machine learning algorithms. Semantic Web data has both common and unique features compared to other general data sets for machine learning. Specifically, Semantic Web data has the following primary dimensions of characteristics. The first one is that one often has to assume an open world. This classification coordinate is also an important one in data quality research on conventional areas. It considers the possibility of two assumptions in the data model, namely: the Closed World Assumption (CWA) and the Open World Assumption (OWA). The CWA assumes, in the context of a logical formulation of the data model, that the data in a schema are all and only the data that satisfy the data schema. On the contrary, the OWA assumes that the data in the schema are a subset of the data satisfying the data schema. I will discuss it more in the background introduction for data quality in

Chapter 2. Because my research is to detect potentially erroneous data, in machine learning theory, whether the data set is noisy or clean is another critical aspect for consideration. For object property triples, I design three systems for different real world Semantic Web data sets according to the dimensions introduced above. I also notice that the data set size is a significant factor on the choice of machine learning algorithms. Since it is a common aspect for designing most of the machine learning algorithms, I consider it in most of the experiments for testing my algorithms rather than design a special algorithm for large data sets.

The rest of the paper is organized as follows. Chapter 2 provides the readers with an overview of various technologies and research areas that I have explored, used and benefited from in this thesis. In this chapter, I discuss various technologies that are the building blocks of the Semantic Web. In addition, I also survey work from related research areas and describe how the work summarized is similar to or different from the work I have done in my thesis. Most of the related research work discussed in this chapter is related to my research as a whole. Other research areas that are related to specific techniques used in my research will be compared and contrasted when I discuss each technique in the thesis. Chapter 3 describes the problem of detecting abnormal Semantic Web data. It provides the skeleton of situations that the algorithms in the following chapters are trying to deal with. Chapter 4 describes the algorithm to evaluate object property triples in the situation where the closed world assumption is valid. The closed world assumption implies that the data is richly described and thus useful patterns in it are relatively easier to learn and discover. Chapter 5 describes the algorithm for the situation where open world assumption is more appropriate. To better deal with the changes on the assumption of unobserved data, the system attempts to collect more contextual information and learn from existing data. Chapter 6 describes the algorithm for noisy data learning. Traditional anomaly detection techniques focus on detecting anomalies in new data after training on normal (or clean) data [29]. However, there are many cases where we cannot find such a clean data for training and we need to detect anomalies directly in a noisy data set. Thus this algorithm is designed for detecting abnormal Semantic Web data in a data set that contains a large number of normal elements and a significant portion of abnormal data as well. Chapter 7 describes a general algorithm for detecting abnormal data in both object property and datatype property triples. This algorithm extends the concept of functional dependency from relational databases onto RDF graphs. It also searches value-clustered graph functional dependency based on finding natural clusters of property values. In each of the chapters 4 to 7, the content is organized in answering the questions from 2 to 4 listed above. The experiments are all described along with each algorithm in these chapters. The last chapter is the conclusion where I analyze my thesis from a critical perspective to identify lessons learned and set directions for future work.

Chapter 2

Background

In this chapter I review important terminology and discuss work related to the thesis. First, I provide a brief introduction to the Semantic Web languages. Next I introduce some background about data quality. Then, I discuss some representative works related to data quality on the Semantic Web.

2.1 Semantic Web Languages

The goal of the Semantic Web is to automate machine processing of web documents by making their meanings explicit. To do this, Semantic Web researchers have developed languages and software that add explicit semantics to the content-structuring aspects of XML. The Semantic Web extends the existing web with structure, and provides a mechanism to specify formal and shareable semantics. A semantic web language allows users to create ontologies [40], which specify standard terms and machine-readable definitions. An ontology is a formal logic based description of a vocabulary that allows one to talk about a domain of discourse. The vocabulary is articulated using definitions and relationships among the defined concepts. As ontologies use formal logic, the intended meaning of assertions using the vocabulary is unambiguous, and therefore, it avoids the ambiguities of natural language. Information resources (such as web pages and databases) then commit to one or more ontologies, thus stating which sets of definitions are applicable. Further, since ontologies are expressed using formal logic, we can use software to ”infer” implicit information in addition to what is explicitly stated. For example, an animals ontology might state that the class Dog is a subclass of Mammal and that the classes Mammal and Fish are disjoint. These definitions communicate some of the meaning of the terms in the resource, and can be used by logical reasoning systems to deduce information that was not explicitly stated.

OWL is an ontology language designed specifically for the Web that is compatible with XML, as well as other W3C standards. Specifically, OWL extends the Resource Description Framework (RDF)[27] and RDF Schema [6], two early Semantic Web standards endorsed by the W3C. Syntactically, an OWL ontology is a valid RDF document and a valid XML document. This allows OWL to be processed by the wide range of XML and RDF tools that are already available. In this section I first describe RDF and RDF Schema which provide a starting point for designing OWL. When introducing RDF, I also discussed our main perspective of RDF in this work. Then I briefly introduce description logic (DL), the logic which OWL is based on, and how OWL incorporates DL to specify semantics for web data.

2.1.1 RDF and RDF schema

Within the RDF data model all objects of interest are called resources. Resources have properties. Each property has a property type and a property value. Property values can be atomic, e.g. strings or numbers, or references to other resources, which in turn may have their own properties. Information about resources is represented in the form of triples. Each triple represents a single property of a resource. Triples can be compared to simple sentences. Each triple consists of a subject, a predicate, and an object. The subject determines the resource which is described by the triple. The predicate determines a property type. The object contains the property value. Triples can be visualized as node and arc diagrams. In this notation, a triple is represented by a node for the subject, a node for the object, and an arc for the predicate, directed from the subject node to the object node. A set of triples forms a directed labeled graph by sharing subjects and objects.

The subject, predicate and object of an RDF triple are RDF nodes. There are three different types of nodes.

1. URI References are nodes that are identified by a globally unique identifier following the URI syntax. URIs can be classified as locators (URLs), as names (URNs), or as both. A uniform resource name (URN) functions like a person's name, while a Uniform Resource Locator (URL) resembles that person's street address. In other words: the URN defines an item's identity, while the URL provides a method for finding it. An example of URI is `http://example.org/wiki/URI#Examples_of_URI_references`. Within RDF, URI references may be used to identify any kind of object, including Web resources such as HTML documents, real world entities such as products, organizations and persons, and abstract concepts such as terms, classes, or property types. The globally unique identification of a resource eases the integration of information about a resource from distinct information providers. Therefore any resource which might be described by multiple information providers should be identified by a URI reference. A URI owner who assigns a URI reference to a resource should provide representations of the resource. This enables information consumers to retrieve authoritative information about resources by dereferencing URIs. For instance, an information consumer might discover an RDF term on the Web that he does not understand. In an attempt to understand the term, he could dereference the term's URI and retrieve a part of the ontology that defines the unknown term and might relate the term to terms which the information consumer understands.

2. Blank Nodes. For identifying resources that do not need to be referenced from outside the RDF graph in which they occur, the RDF data model provides blank nodes as a second, alternative identification mechanism. Blank nodes are unique nodes that can be used in one or more RDF triples to identify a resource. The term "blank" refers to the fact that blank nodes do not have identifiers. It is only possible to determine whether two blank nodes are the same or not. Within implementations of the RDF data model, blank nodes are

often assigned local identifiers for practical reasons. These identifiers do not have any meaning at the data model level. As blank node identifiers are only unique within the scope of the graph in which they occur, it is possible that distinct blank nodes in different graphs use the same blank node identifier. For instance, a blank node with the identifier BN1 might be used within one RDF graph to refer to Bob's address. A different blank node, which also uses the identifier BN1, might be used in another graph to identify Peter's address. In order to avoid confusion between Bob's and Peter's addresses and to preserve the meaning of both graphs, their blank nodes must be kept distinct. Thus, when RDF graphs are merged within implementations, it might be necessary to rename blank nodes in order to avoid collisions.

3. Literals are used to represent property values such as text, numbers, and dates. Literals may be plain or typed. A plain literal is a string combined with an optional language tag. The language tag identifies a natural language, such as English or Chinese. A typed literal is a string combined with a datatype URI. The datatype URI identifies the datatype of the literal. Datatype URIs for common datatypes such as integers, floating point numbers and dates are defined by the XML Schema datatypes specification [1].

Consider, for example, that we want to say Jeff advises Yang. This statement will be represented in an RDF graph with a source that denotes Jeff, a directed edge from source to destination that denotes the advises relationship and a destination that denotes Yang. In RDF we need URIs (or URLs) to refer to Jeff and the advises relationship. Yang can be either a literal or a URI. The following is one version of the RDF graph represented in XML syntax. In the example below, the xmlns attribute in the <RDF> tag gives the rdf: and ex: prefixes a qualified namespace. XML Namespaces provide a method to avoid element name conflict. The namespace is defined by the xmlns attribute in the start tag of an element. The namespace declaration has the following syntax. xmlns:prefix="URI". When a namespace is

[1] http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/

defined for an element, all child elements with the same prefix are associated with the same namespace.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ex="http://example.com/"
>
<rdf:Description rdf:about="http://example.com/jeff">
    <ex:advises>Yang</ex:advises>
</rdf:Description>
</rdf:RDF>
```

RDF Schema (RDFS) extends the RDF vocabulary to allow describing taxonomies of classes and properties. All resources can be divided into groups called classes. Classes are also resources, so they are identified by URIs and can be described using properties. The members of a class are instances of classes, which is stated using the *rdf:type* property, and the set of these instances is called the class extension. *rdfs:domain* is an instance of *rdf:Property* that is used to state that any resource that has a given property is an instance of one or more classes. A triple of the form: P *rdfs:domain* C states that P is an instance of the class *rdf:Property*, that C is an instance of the class *rdfs:Class* and that the resources denoted by the subjects of triples whose predicate is P are instances of the class C. *rdfs:range* is an instance of *rdf:Property* that is used to state that the values of a property are instances of one or more classes. The triple P *rdfs:range* C states that P is an instance of the class *rdf:Property*, that C is an instance of the class *rdfs:Class* and that the resources denoted by the objects of triples whose predicate is P are instances of the class C. The property *rdfs:subClassOf* is an instance of *rdf:Property* that is used to state that all the instances of one class are instances of another. A triple of the form: $C1$ *rdfs:subClassOf* $C2$ states that $C1$ is an instance of *rdfs:Class*, $C2$ is an instance of *rdfs:Class* and $C1$ is a subclass of $C2$. The property *rdfs:subPropertyOf* is an instance of *rdf:Property* that is used to state that all resources related by one

Table 2.1: RDFS classes.

Element	Class of	rdfs:subClassOf	rdf:type
rdfs:Resource	all resources	rdfs:Resource	rdfs:Class
rdfs:Class	all classes	rdfs:Resource	rdfs:Class
rdfs:Literal	literal values	rdfs:Resource	rdfs:Class
rdfs:Datatype	datatypes	rdfs:Class	rdfs:Class
rdf:XMLLiteral	XML literal values	rdfs:Literal	rdfs:Datatype
rdf:Property	properties	rdfs:Resource	rdfs:Class
rdf:Statement	statements	rdfs:Resource	rdfs:Class
rdf:List	lists	rdfs:Resource	rdfs:Class
rdfs:Container	containers	rdfs:Resource	rdfs:Class
rdf:Bag	unordered containers	rdfs:Container	rdfs:Class
rdf:Seq	ordered containers	rdfs:Container	rdfs:Class
rdf:Alt	containers of alternatives	rdfs:Container	rdfs:Class
rdfs:Container MembershipProperty	rdf:_1... properties expressing membership	rdf:Property	rdfs:Class

property are also related by another. A triple of the form: $P1$ *rdfs:subPropertyOf* $P2$ states that $P1$ is an instance of *rdf:Property*, $P2$ is an instance of *rdf:Property* and $P1$ is a subproperty of $P2$. The list of classes defined by RDFS is shown in the Table 2.1.1.

2.1.2 RDF Graph

Because the Semantic Web has two languages: OWL and RDF, and an OWL ontology is a valid RDF document, this thesis mainly focus on Semantic Web data in RDF forms. In this work, we will mainly investigate RDF data as a graph-based data model. RDF is closely related to semantic networks [81]. Semantic networks are a well-known and very flexible knowledge representation mechanism. Similar to semantic networks, RDF statements can be expressed in a graph with labeled nodes connected by directed and labeled edges. Essentially, the subject of a RDF statement is the source node of the edge, the object is the target node of the edge and the edge is the predicate relating the subject and the object. For example, the following piece of RDF statements can be modeled as Figure 2.1.

```
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:foaf="http://xmlns.com/foaf/0.1/"
 xmlns="http://www.example.org/˜joe/contact.rdf#">
```

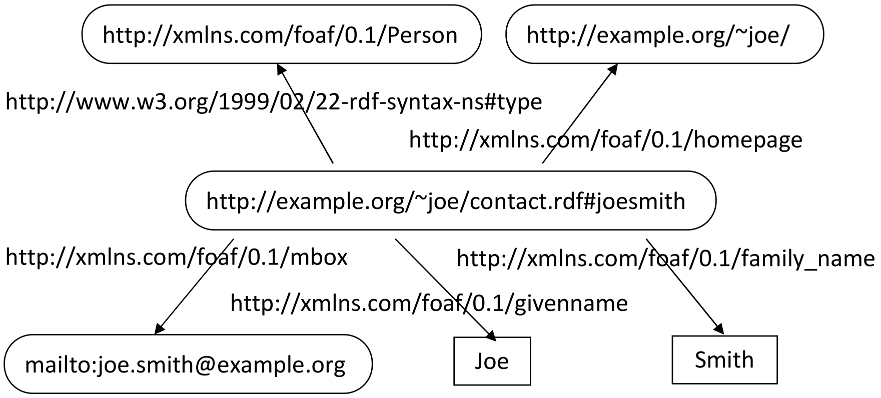

http://xmlns.com/foaf/0.1/Person
http://example.org/~joe/
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://xmlns.com/foaf/0.1/homepage
http://example.org/~joe/contact.rdf#joesmith
http://xmlns.com/foaf/0.1/mbox
http://xmlns.com/foaf/0.1/family_name
http://xmlns.com/foaf/0.1/givenname
mailto:joe.smith@example.org
Joe
Smith

Figure 2.1: RDF graph describing Joe Smith.

```
<foaf:Person rdf:about=
 "http://www.example.org/˜joe/contact.rdf#joesmith">
    <foaf:mbox rdf:resource="mailto:joe.smith@example.org"/>
    <foaf:homepage
     rdf:resource="http://www.example.org/˜joe/"/>
    <foaf:family_name>Smith</foaf:family_name>
    <foaf:givenname>Joe</foaf:givenname>
</foaf:Person>
</rdf:RDF>
```

After this example, I formally define some of the terminology that will be often used in this work.

Definition 1. *An RDF graph is $G := (I, L, R, E)$, where three sets I, L and R are instance, literal and property identifiers and the set of directional edges is $E \subseteq I \times R \times (I \cup L)$. An edge $e \in E$ of form $< s, p, o >$ has the start point $s \in I$, end point $o \in (I \cup L)$ and the label $r \in R$ on it. Let $\mathcal{G}$ be the set of all possible graphs and $G \in \mathcal{G}$. Let $R^- = \{r^- | r \in R\}$.*

In this definition, I listed the parts constituting an RDF graph. More importantly, I point out the inverse property r^-, given the property r. As I introduced in section 2.1.1, properties have a direction, from domain to range. In practice, people

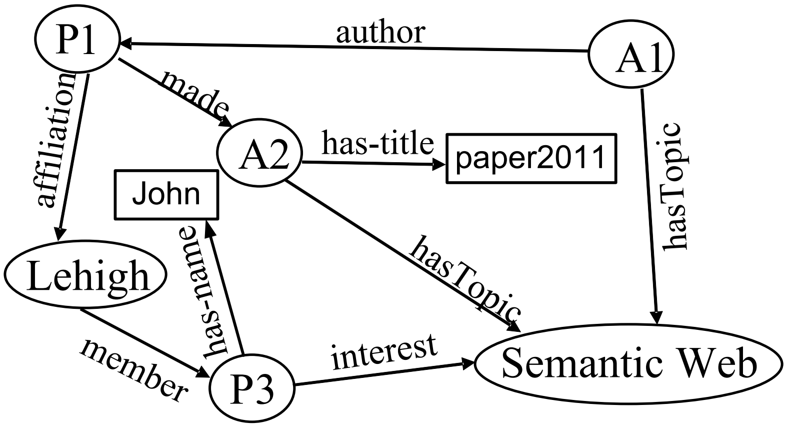


Figure 2.2: Examples of definitions.

often find it useful to define relations in both directions: persons own cars, cars are owned by persons. Whenever we have a triple using the property of one direction, we can easily create an equivalent triple by using the inverse of original property and flipping the original subject and object.

Definition 2. *Given an RDF graph* $G := (I, L, R, E)$*, a Path* c *in graph* G *is a tuple* $< I_0, r_1, I_1, ..., r_n, I_n >$ *where* $I_i \in I, r_i \in R \cup R^-$*, and* $\forall i, 0 \leq i < n$*, if* $r_i \in R$ *then* $< I_i, r_{i+1}, I_{i+1} > \in E$ *or if* $r_{i+1} \in R^-$ *then* $< I_{i+1}, r_{i+1}, I_i > \in E$*;* $\forall j$*, if* $i \neq j$ *then* $I_i \neq I_j$. $Edges(c) = \{r_1, r_2, ..., r_n\}$. $Length(c) = n$. $First(c) = I_0$. $Last(c) = I_n$.

Paths are acyclic and directional, but can include inverted relations of the form r^-.

Definition 3. *Given an RDF graph* $G := (I, L, R, E)$*, the set of all Paths from node* I_0 *to* I_n *is defined as:* $Paths(I_0, I_n, G) = \{p_i | p_i \text{ is a Path}, First(p_i) = I_0 \text{ and } Last(p_i) = I_n\}$.

To make our definitions clear, I give another example RDF graph (Fig. 2.2). In this figure, the circles represent instances (I); the boxes represent literals (L); the labels on the edges are relation identifiers (R) and the arcs between circles and boxes are edges (E). An example of Path (Definition 2) in this RDF graph is $< P1, affiliation, Lehigh, member, P3 >$. Note as stated in Definition 2,

$< A2,\ made^{-},\ P1,\ author^{-},\ A1,\ hasTopic,\ SemanticWeb >$ is also a Path on which some inverse properties are involved. Thus the set of Paths between two nodes $A2$ and $SemanticWeb$ include the one shown above, the Path $< A2,\ made^{-},\ P1,\ affiliation,\ Lehigh,\ member,\ P3,\ interest,\ SemanticWeb >$ and the edge connecting them directly.

2.1.3 RDF Query Language SPARQL

Given a set of RDF data, one of the best ways to explore the data is through SPARQL (pronounced "sparkle"), an RDF query language. Its name is a recursive acronym that stands for SPARQL Protocol and RDF Query Language. It was standardized by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium, and is considered a key semantic web technology. On 15 January 2008, SPARQL became an official W3C Recommendation. SPARQL allows users to write globally unambiguous queries. For example, the following query returns a person and the name of another he/she knows in the world:

```
PREFIX advises: <http://www.example.com/exampleOntology#advises>
PREFIX researchTopic:
       <http://www.example.com/exampleOntology#researchTopic>
SELECT ?x ?name
WHERE {
 ?x advises ?y
 ?y researchTopic ?topicName
}
```

The example is assuming the ontology `http://www.example.com/exampleOntology` to describe persons and their advises relationships. This illustrates the Semantic Web's vision of treating the Web as a single enormous database. This query can be distributed to multiple SPARQL endpoints, computed distributedly, and results gathered, a procedure known as federated query. We will explain this query by introducing the preliminary blocks first.

SPARQL closely resembles SQL and offers a relational, pattern-based approach to retrieving data from a store. Consider a graph, G contains triples that share objects or subjects (listed below).

```
  James  advises  Jeff
  Jeff advises  Yang
  Jeff researchTopic   "Artificial Intelligence"
  Yang  researchTopic   "Intelligent Agent"
  Yang  researchTopic   "Semantic Web"
```

SPARQL's approach to selecting values is to take triples and allow them to contain variables (denoted by a ? or \$ before a string). These structures - triple patterns - match against real triples in the store, or inferred triples if you wish to use a reasoner. Every time a triple pattern matches against a triple, it produces a binding for each variable. For example, the triple pattern *James advises ?y* produces one binding for ?y: Jeff. The pattern *?x advises ?y* produces a richer table of bindings:

```
  |   x   |   y   |
  =================
1 | James |  Jeff |
-------------------
2 | Jeff  |  Yang |
```

Each row in this table is a result for the query. Variables can also occur in multiple patterns that together comprise a query. Patterns that overlap in variables narrow down the results, while those that do not expand them. To return the earlier example SPARQL query, the pattern *?x advises ?y* and *?y researchTopic ?topicName* produces the following results:

```
  |   x   |   y   |              name            |
  ================================================
1 | James | Jeff   |  "Artificial Intelligence"  |
-------------------------------------------------
```

```
2 | Jeff   | Yang   |     "Intelligent Agent"     |
-------------------------------------------------
3 | Jeff   | Yang   |     "Semantic Web"          |
```

From the above results, we can see that a row exists in the results for every possible substitution of values into the query that would yield a set of triples that exist in the graph. Each row can contain only one binding, so Yang's two research topics fork the results.

Now let's break down the earlier query into its parts to better understand the syntax. Starting from the top we encounter the PREFIX keyword. PREFIX is essentially the SPARQL equivalent of declaring an XML namespace: it associates a short label with a specific URI. And, just like a namespace declaration, the label applied carries no particular meaning. It's just a label. A query can include any number of PREFIX statements. The label assigned to a URI can be used anywhere in a query in place of the URI itself; for example, within a triple pattern. In the single triple pattern included in this query we can see the *advises* prefix in use as a shorthand for `http://www.example.com/exampleOntology\#advises`, the full URI of a property in the example ontology. The start of the query proper is the SELECT keyword. Like its twin in a SQL query, the SELECT clause is used to define the data items that will be returned by a query. The key function of the WHERE clause is to describe a graph pattern which is a collection of triple patterns, as introduced above, that identify the shape of the graph that we want to match against.

2.1.4 OWL

RDF is a simple data model and as such it does not have any significant semantics. RDF Schema addresses this shortcoming by allowing the user to define a vocabulary consisting of rdfs:Class, rdf:Property, rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain, and rdfs:range for use in RDF models. OWL extends RDF and RDFS. Its primary aim is to bring the expressivity and reasoning power of description logic to the semantic web. Each of the important RDF Schema terms are either included directly in OWL or are superseded by new OWL terms. Unfortunately, not everything from RDF can be expressed in DL. For example, the classes of classes are not permitted in DL, and some of the triple expressions would have no sense in DL. That is why OWL can be only syntactic extension of RDF/RDFS (note that RDFS is both a syntactic and semantic extension of RDF). To partially overcome this problem, and also to allow layering within OWL, three species of OWL are defined.

OWL Lite can be used to express taxonomy and simple constraints, such as 0 and 1 cardinality. It is the simplest OWL language and corresponds to the description logic SHIF. SHIF consists of the basic DL Attributive Language with Complements (ALC), plus transitivity, hierarchical roles, inverse roles, and functional roles. OWL DL supports maximum expressiveness while retaining computational completeness and decidability. The DL in the name shows that it is intended to support description logic capabilities. OWL DL corresponds to the description logic SHOIN which includes ALC, plus transitive roles, hierarchical roles, nominals, inverse roles and cardinality restrictions. OWL Full has no expressiveness constraints, but also does not guarantee any computational properties. It is formed by the full OWL vocabulary, but does not impose any syntactic constraints, so that the full syntactic freedom of RDF can be used.

These three languages are layered in a sense that every legal OWL Lite ontology is a legal OWL DL ontology, every legal OWL DL ontology is a legal OWL Full ontology, every valid OWL Lite conclusion is a valid OWL DL conclusion, and every valid OWL DL conclusion a valid OWL Full conclusion. The inverses of these relations generally do not hold. Also, every OWL ontology is a valid RDF document (i.e., DL expressions are mapped to triples), but not all RDF documents are valid OWL Lite or OWL DL documents.

OWL classes are described through ”class descriptions”, which can be combined into ”class axioms”. I first describe class descriptions and subsequently turn to class axioms. Two OWL class identifiers are predefined, namely the classes owl:Thing and owl:Nothing. The class extension of owl:Thing is the set of all individuals. The class extension of owl:Nothing is the empty set.

OWL distinguishes six types of class descriptions:

1. A class identifier (a URI reference).

2. An exhaustive enumeration of individuals that together form the instances of a class. A class description of the ”enumeration” kind is defined with the owl:oneOf property. The value of this built-in OWL property must be a list of individuals that are the instances of the class. This enables a class to be described by exhaustively enumerating its instances. The class extension of a class described with owl:oneOf contains exactly the enumerated individuals, no more, no less.

3. A property restriction. It describes an anonymous class, namely a class of all individuals that satisfy the restriction. OWL distinguishes two kinds of property restrictions: value constraints and cardinality constraints. A value constraint puts constraints on the range of the property when applied to this particular class description. A cardinality constraint puts constraints on the number of values a property can take, in the context of this particular class description.

4. The intersection of two or more class descriptions. An owl:intersectionOf statement describes a class for which the class extension contains precisely those individuals that are members of the class extension of all class descriptions in the list.

5. The union of two or more class descriptions. An owl:unionOf statement describes an anonymous class for which the class extension contains those individuals that occur in at least one of the class extensions of the class descriptions in the list.

6. The complement of a class description. An owl:complementOf statement describes a class for which the class extension contains exactly those individuals that do not belong to the class extension of the class description that is the object of the statement. owl:complementOf is analogous to logical negation:

Table 2.2: OWL DL axioms and facts.

Abstract Syntax	DL syntax	Semantics
Classes		
Class(A partial $C_1 ... C_n$)	$A \sqsubseteq C_1 \sqcap ... \sqcap C_n$	$A^{\mathcal{I}} \subseteq C_1^{\mathcal{I}} \cap ... \cap C_n^{\mathcal{I}}$
Class(A complete $C_1 ... C_n$)	$A \equiv C_1 \sqcap ... \sqcap C_n$	$A^{\mathcal{I}} \equiv C_1^{\mathcal{I}} \cap ... \cap C_n^{\mathcal{I}}$
EnumeratedClass($A\ o_1 ... o_n$)	$A \equiv \{o_1, ..., o_n\}$	$A^{\mathcal{I}} = \{o_1^{\mathcal{I}}, ..., o_n^{\mathcal{I}}\}$
SubClassOf($C_1\ C_2$)	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \sqsubseteq C_2^{\mathcal{I}}$
EquivalentClasses($C_1\ ...\ C_n$)	$C_1 \equiv ... \equiv C_n$	$C_1^{\mathcal{I}} \equiv ... \equiv C_n^{\mathcal{I}}$
DisjointClasses($C_1\ ...\ C_n$)	$C_i \sqcap C_j = \bot, i \neq j$	$C_i^{\mathcal{I}} \cap C_j^{\mathcal{I}} = \emptyset, i \neq j$
Datatype(D)		$D^C \triangle_D^{\mathcal{I}}$
Datatype Properties		
DatatypeProperty(		
$U\ super(U_1 ... super(U_n)$	$U \sqsubseteq U_i$	$U^{\mathcal{I}} \subseteq U_i^{\mathcal{I}}$
$domain(C_1) ... domain(C_m)$	$\geqslant 1\ U \sqsubseteq C_i$	$U^{\mathcal{I} \subseteq C_i^{\mathcal{I}} \times \triangle_D^{\mathcal{I}}}$
$range(D_1) ... range(D_{l)}$	$\top \sqsubseteq \forall U.D_i$	$U^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times D_i^{\mathcal{I}}$
[Functional])	$\top \sqsubseteq \leqslant\ 1U$	U_i is functional
$SubPropertyOf(U_1\ U_2)$	$U_1 \sqsubseteq U_2$	$U_1^{\mathcal{I}} \subseteq U_2^{\mathcal{I}}$
$EquivalentProperties(U_1 ... U_n)$	$U_1 \equiv ... \equiv U_n$	$U_1^{\mathcal{I}} \equiv ... \equiv U_n^{\mathcal{I}}$
Object Properties		
ObjectProperty(		
$R\ super(R_1 ... super(R_n)$	$R \sqsubseteq R_i$	$R^{\mathcal{I}} \subseteq R_i^{\mathcal{I}}$
$domain(C_1) ... domain(C_m)$	$\geqslant 1\ R \sqsubseteq C_i$	$R^{\mathcal{I} \subseteq C_i^{\mathcal{I}} \times \triangle_D^{\mathcal{I}}}$
$range(C_1) ... range(C_{l)}$	$\top \sqsubseteq \forall R.D_i$	$R^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times D_i^{\mathcal{I}}$
$[inverseOf(R_0)]$	$R \equiv (R_0^-)$	$R^{\mathcal{I}} = (R_0^{\mathcal{I}})^-$
$[Symmetric]$	$R \equiv (R^-)$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^-$
$[Functional]$	$\top \sqsubseteq \leqslant\ 1R$	$R^{\mathcal{I}}\ is\ functional$
$[InverseFunctional]$	$\top \sqsubseteq \leqslant\ 1R^-$	$(R^{\mathcal{I}})^-\ is\ functional$
$[Transitive]$)	$Tr(R)$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^+$
$SubPropertyOf(R_1\ R_2)$	$R_1 \sqsubseteq R_2$	$R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$
$EquivalentProperties(R_1 ... R_n)$	$R_1 \equiv ... \equiv R_n$	$R_1^{\mathcal{I}} \equiv ... \equiv U_n^{\mathcal{I}}$
Annotation		
$AnnotationProperty(S)$		
Individuals		
$Individual($		
$o\ type(C_1) ... type(C_n)$	$o \in C_i$	$o^{\mathcal{I}} \in C_i^{\mathcal{I}}$
$value(R_1\ o_1) ... value(R_n\ o_n)$	$o, o_i \in R_i$	$\{o^{\mathcal{I}}, o_i^{\mathcal{I}}\} \in R_i^{\mathcal{I}}$
$value(U_1\ v_1) ... value(U_n\ v_n))$	$o, v_i \in U_i$	$\{o^{\mathcal{I}}, v_i^{\mathcal{I}}\} \in U_i^{\mathcal{I}}$
$SameIndividual(o_1 ... o_n)$	$o_1 = ... = o_n$	$o_i^{\mathcal{I}} = ... = o_n^{\mathcal{I}}$
$DifferentIndividual(o_1 ... o_n)$	$o_i \neq o_j, i \neq j$	$o_i^{\mathcal{I}} \neq o_j^{\mathcal{I}}, i \neq j$

the class extension consists of those individuals that are NOT members of the class extension of the complement class.

Class axioms typically contain additional components that state necessary and/or sufficient characteristics of a class. OWL contains three language constructs for combining class descriptions into class axioms. $rdfs : subClassOf$ allows one to say that the class extension of a class description is a subset of the class extension of another class description. $owl : equivalentClass$ allows one to say that a class description has exactly the same class extension as another class description. $owl : disjointWith$ allows one to say that the class extension of a class description has no members in common with the class extension of another class description.

OWL distinguishes between two main categories of properties that an ontology builder may want to define: object properties that link individuals to individuals and datatype properties that link individuals to data values. An object property is defined as an instance of the built-in OWL class owl:ObjectProperty. A datatype property is defined as an instance of the built-in OWL class owl:DatatypeProperty.

More detailed OWL DL axioms and facts are summarized in Table 2.2. Elementary descriptions are atomic concepts and atomic roles. Complex descriptions can be built from them inductively with concept constructors. In abstract notation, I use the letters A and B for atomic concepts, the letter R for atomic roles, and the letters C and D for concept descriptions. Concept descriptions in DL are formed according to the following syntax rule:

$$\begin{array}{lll} C, D \rightarrow & A \mid & (atomic\ concept) \\ & \top \mid & (universal\ concept) \\ & \bot \mid & (bottom\ concept) \\ & \neg\ A \mid & (atomic\ negation) \\ & C\ \sqcap D \mid & (intersection) \\ & \forall\ R.C \mid & (value\ restriction) \\ & \exists\ R.\top \mid & (limited\ existential\ quantification). \end{array}$$

To define a formal semantics of DL, we consider interpretations $\mathcal{I}$ that consist of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of the interpretation) and an interpretation function, which assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is extended to concept descriptions by the following inductive definitions:

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \ \cap \ D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} | \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} | \exists b.(a,b) \in R^{\mathcal{I}}\}
\end{aligned}$$

The above features are provided since OWL 1. OWL 1 was mainly focused on constructs for expressing information about classes and individuals, and exhibited some weakness regarding expressiveness for properties. In 2009, OWL 2 added some new features. For example OWL 2 offers new constructs for expressing additional restrictions on properties, new characteristics of properties, incompatibility of properties, property chains and keys. I will introduce property composition that can be useful for this thesis. OWL 1 does not provide a means to define properties as a composition of other properties, as uncle could be defined as brother of father. Thus, it is not possible to propagate a property (e.g.; locatedIn) along another property (e.g.; partOf). The OWL 2 construct $ObjectPropertyChain$ in a $SubObjectPropertyOf$ axiom allows a property to be defined as the composition of several properties. For example $locatedIn \circ partOf \sqsubseteq locatedIn$ means that if x is located in y and y is part of z then x is located in z, for example a disease located in a part is located in the whole. I use the notation $\circ$ to represent a composition between two properties in this thesis.

2.2 Data Quality and Data Cleansing

The terms Information Quality (IQ) and Data Quality (DQ) are often used interchangeably and in this work we focus on the technical aspects, so we do not make explicit distinction between them. Data quality has serious consequences, of far-reaching significance, for the efficiency and effectiveness of organizations and businesses. In its report on data quality, the Data Warehousing Institute ([27]) estimates that data quality problems cost U.S. businesses more than 600 billion dollars a year. To be processable and interpretable in a effective and efficient manner, data has to satisfy a set of quality criteria. I first give some commonly recognized data quality dimensions. Then I discuss several methodologies that have been developed before that provide a rationale for the optimal choice of such activities and techniques. Finally, more detailed technical analysis of data cleansing approaches are discussed.

2.2.1 Data Quality Dimensions

Data quality has the following major dimensions: accuracy, completeness and consistency. Besides the major ones, time-related dimensions are currency, timeliness and volatility. In this work, I mainly focus on the errors on the Semantic Web that can be categorized into major data quality dimensions.

Accuracy is defined as the closeness between a value v and a value v', considered as the correct representation of the real-life phenomenon that v aims to represent. As an example if the name of a person is John, the value $v' = John$ is correct, while the value $v = Jhn$ is incorrect. Two kinds of accuracy can be identified, namely *syntactic accuracy* and *semantic accuracy*. Syntactic accuracy is the closeness of a value v to the elements of the corresponding definition domain D. In syntactic accuracy we are not interested in comparing v with the true value v'; rather, we are interested in checking whether v is any one of the values in D, whatever it is. So, if $v = Jack$, even if $v' = John$, v is considered syntactically correct, as Jack is an admissible value in the domain of persons' names. Syntactic accuracy is measured by means of functions, called comparison functions, that evaluate the distance between v and the values in D. Edit distance is a simple example of a comparison function, taking into account the minimum number of character insertions, deletions, and replacements to convert a string s to a string s'. More complex comparison functions exist, for instance taking into account similar sounds or character transpositions. *Semantic accuracy* is the closeness of the value v to the true value v'. For example, if we say the director of movie Casablanca is Cameron, then it is a semantic error. Because Cameron is an real director in this domain, it is an admissible value and so would not be considered as a syntactic error. However Curtiz is the director of Casablanca and then it is a semantic error. Note that, while it is reasonable to measure syntactic accuracy using a distance function, semantic accuracy is measured better with a $< yes, no >$ or a $< correct, incorrect >$ domain. Consequently, semantic accuracy coincides with the concept of correctness. In contrast with what happens for syntactic accuracy, in order to measure the semantic accuracy of a value v, the corresponding true value has to be known, or, else, it should be possible, considering additional knowledge, to deduce whether the value v is or is not the true value. From the above arguments, it is clear that semantic accuracy is typically more complex to calculate than syntactic accuracy. When it is known a priori that the rate of errors is low, and the errors result typically from typos, then syntactic accuracy tends to coincide with semantic accuracy, since typos produce values close to the true ones. As a result, semantic accuracy may be achieved by replacing an inaccurate value with the closest value in the definition domain, under the assumption that it is the true one.

Completeness can be generically defined as "the extent to which data is of sufficient breadth, depth, and scope for the task at hand" [96]. Three types of completeness are identified by Pipino et al. [79]. Schema completeness is defined as the degree to which entities and attributes are not missing from the schema. Column completeness is defined as a measure of the missing values for a specific property or column in a table. Population completeness evaluates missing values with respect to a reference population. In the following we refer to the relational model. Intuitively, the completeness of a table characterizes the extent to which the table represents the corresponding real world. Completeness in the relational model can be characterized with respect to: (i) the presence/absence and meaning of null values, and (ii) the validity of one of the two assumptions called the open world assumption and closed world assumption [85]. In order to characterize completeness, it is important to understand why a model has null values, i.e. the value is missing. Indeed, a value can be missing either because it exists but is unknown, or because it does not exist at all, or because it may exist but it is not actually known whether it exists or not. The three types of null values are not existing, existing but unknown, not known if existing [9]. In logical models for databases, such as the relational model, there are two different assumptions on the completeness of data represented in a relation instance r. The closed world assumption (CWA) states that only the values actually present in a relational table r, and no other values represent facts of the real world. In the open world assumption (OWA) we can state neither the truth nor the falsity of facts not represented in the tuples of r. As I briefly introduced in the introduction, these two assumptions also are important effects on the Semantic Web and so our consideration of system design takes them into account as well.

The consistency dimension captures the violation of semantic rules defined over (a set of) data items, where items can be tuples of relational tables or records in a file. With reference to relational theory, *integrity constraints* are an instantiation of such semantic rules. In statistics, *data edits* are another example of semantic rules that allow for the checking of consistency. Integrity constraints are properties that must be satisfied by all instances of a database schema. Although integrity constraints are typically defined on schemas, they can at the same time be checked on a specific instance of the schema that presently represents the extension of the database. Therefore, we may define integrity constraints for schemas, describing a schema quality dimension, and for instances, representing a data dimension. Most integrity constraints are dependencies. The following are main types of dependencies.

- Key Dependency. This is the simplest type of dependency. Given a relation instance r, defined over a set of attributes, we say that for a subset K of the attributes, a key dependency holds in r, if no two rows of r have the same K-values. For instance, an attribute of social security number can serve as a key in any relation instance of a relation schema $Person$. When key dependency constraints are enforced, no duplication will occur within the relation.

- Inclusion Dependency. Inclusion dependency is a very common type of constraint, and is also known as referential constraint. An inclusion dependency over a relational instance r states that some columns of r are contained in other columns of r or in the instances of another relational instance s. A foreign key constraint is an example of inclusion dependency, stating that the referring columns in one relation must be contained in the primary key columns of the referenced relation.

- Functional Dependency. Given a relational instance r, let X and Y be two nonempty sets of attributes in r. The functional dependency $X \rightarrow Y$ is satisfied in r, if the following holds: for every pair of tuples t_1 and t_2 in r, if $t_1.X = t_2.X$, then $t_1.Y = t_2.Y$, where the notation $t_1.X$ means the projection of the tuple t_1 onto the attributes in X.

Integrity constraints discussed above are within the relational model as a specific category of consistency semantic rules.

However, where data is not relational, consistency rules can still be defined. As an example, in the statistical field, data coming from census questionnaires have a structure corresponding to the questionnaire schema. The semantic rules are thus defined over such a structure in a way very similar to relational constraints. rules. Data edits are example of semantic rules that allow for the checking of consistency. Data edits are less powerful than integrity constraints because they do not rely on a data model like the relational one. Nevertheless, data editing has been done extensively in the national statistical agencies since the 1950s, and has revealed a fruitful and effective area of application. Data editing is defined as the task of detecting inconsistencies by formulating rules that must be respected by every correct set of answers. Such rules are expressed as edits, which denote error conditions. As an example, an inconsistent answer to a questionnaire can be to declare marital status = "married", age = "5 years old". The rule to detect this kind of errors could be the following: if marital status is married, age must not be less than 14. The rule can be put in the form of an edit, which expresses the error condition, namely, marital status = married and age < 14.

An important aspect of data is their change and update in time. The currency dimension concerns how promptly data is updated. If the residential address of a person is updated, i.e. it corresponds to the address where the person lives, then the currency is high. Volatility characterizes the frequency with which data vary in time. For instance, stable data such as birth dates have volatility equal to 0, as they do not vary at all. Conversely, stock quotes, a kind of frequently changing data, have a high degree of volatility due to the fact that they remain valid for very short time intervals. Timeliness expresses how current data is for the task at hand. The timeliness dimension is motivated by the fact that it is possible to have current data that are actually useless because they are late for a specific usage. For instance, the timetable for university courses can be current by containing the most recent data, but it cannot be timely if it is available only after the start of the classes.

2.2.2 Data Quality Methodologies

Data quality methodologies are often defined as a set of guidelines and techniques that, starting from the input information concerning a given reality of interest, defines a rational process for using the information to measure and improve the quality of data of an organization through given phases and decision points. Data quality methodologies may be classified according to several criteria:

1. Data-driven vs. process-driven. This classification is related to the general strategy chosen for the improvement process. Data-driven strategies are based on using data sources exclusively to improve the quality of data; they make use of the data quality activities. In process-driven strategies, the data production process is analyzed and possibly modified to identify and remove the root causes of quality problems. General purpose methodologies may adopt both data-driven and process-driven strategies, with different depth according to the specific methodology.

2. Measurement vs. improvement. Methodologies are needed for measuring / assessing the quality of data, or to improve their quality. Measurement and improvement activities are closely interrelated, since only when DQ measurements are available, is it possible to conceive techniques to be applied and priorities to be established. As a consequence, the boundary between the

methodologies for measurement and improvement is sometimes vague. In the following, we will use the term measurement when we address the issue of measuring the values of a set of data quality dimensions in a database (or a set of databases). We use the term assessment or benchmarking when such measurements are compared to reference values, to enable a diagnosis of the quality of the database.

3. General-purpose vs. special-purpose. A general-purpose methodology covers a wide spectrum of phases, dimensions, and activities, while a special purpose methodology is focused on a specific activity (e.g., measurement, object identification), on a specific data domain (e.g., a census, a registry of addresses of persons), or specific application domains (e.g., biology).

4. Intraorganizational vs. interorganizational. The measurement and improvement activity concerns a specific organization, or a specific sector of the organization, or even a specific process or database. Otherwise, it concerns a group of organizations (e.g., a group of public agencies) cooperating for a common goal (e.g., in the case of public agencies, providing better services to citizens and businesses).

Based on these criteria, the methods that this work primarily focuses on can be defined as follows. First it is data-driven, because I do not put any restriction or assumption of how the data set is generated or which section of data flow the data set is in. Second, I mainly focus on measurement. Although sometimes when my system reports an erroneous triple and the way to correct it is obvious, the system is assessed by the performance of detecting erroneous triples instead of correcting them. Third, my system is designed for general purpose use, because the system is to deal all kinds of Semantic Web data from different domains. Fourth, because my system tries to deal with data set integrated or contributed from different organizations (e.g. DBpedia), it concerns multiple organizations.

2.2.3 Data Cleansing

In the previous subsections, I have introduced the data quality dimensions and the general techniques about it. In this subsection, I will discuss the detailed solutions for it, i.e. data cleansing, data cleaning, or data scrubbing. The term refers to identifying incomplete, incorrect, inaccurate, irrelevant, etc. parts of the data and then replacing, modifying, or deleting this dirty data. The techniques mainly can be categorized as follows.

1. Looking up: Fixing incorrect information such as the postcode matching the suburb is usually done by comparing each record to the correct values in another table. For example, to correct all the postcodes in the data, assuming that the suburb entered is correct, we would write SQL code that would compare the postcode of the record against a table of postcode + suburb + state that we may have obtained. Such a process would likely generate a list of records where the suburb was not found, requiring a manual investigation and correction of the data.

2. Parsing: Parsing in data cleansing is performed for the detection of syntax errors. A parser decides whether a string of data is acceptable within the allowed data specification. This is similar to the way a parser works with grammars and languages.

3. Data transformation: Data transformation allows the mapping of the data from its given format into the format expected by the appropriate application. This includes value conversions or translation functions, as well as normalizing numeric values to conform to minimum and maximum values. Correcting the formatting of the data is usually done using some pretty simple SQL perhaps combined with logic programming. Users need to decide the format they wish to apply to the data, for example, whether they would like the suburb in title case or all capitals. While this is much less important than getting the data actually right, it can help to make the communications look more professional.

4. Duplicate elimination: Duplicate detection requires an algorithm for determining whether data contains duplicate representations of the same entity. Usually, data is sorted by a key that would bring duplicate entries closer together for faster identification. If the duplicates have some identical values for selected properties, finding duplicates is a fairly easy task for someone who knows a little about the SQL database language. However it is more difficult to find similar records that really are the same real world object, but are not listed in exactly the same way in the database. For instance the following two records may actually be the same person:

   ```
   ID  |First name|Surname| Address1    | Suburb   |Postcode|State
   3442| John     |Citizen| PO Box 33   | Frankston| 3199   |VIC
   682 | Jonathon |Citien |14 Beach Road| FRANKSTON| 3199   |VIC
   ```

   Finding records such as the above calls for what is usually called “Fuzzy Matching”. Because we cannot confidently use logic to determine whether or not two records are the same in the case given above, usually fuzzy matching would leave the data as is, but produce an exception report, highlighting likely duplicate records. Although it is possible to set up customized de-duplication process to remove all the duplicates and clean up all the records automatically, users typically prefer to manually process the data cleanup to ensure that only the correct data is kept, and that all associated pieces of information are transferred across to the valid record e.g. customer payment history.

5. Statistical methods: By analyzing the data using the values of mean, standard deviation, range, or clustering algorithms, it is possible for an expert to find values that are unexpected and thus erroneous. Although the correction of such data is difficult since the true value is not known, it can be resolved by setting the values to an average or other statistical value. Statistical methods can also be used to handle missing values which can be replaced by one or more plausible values, which are usually obtained by extensive data augmentation algorithms.

Of the above popular methods used to clean data, statistical methods are the most general and difficult. Below is a set of general methods that can be utilized for statistical error detection:

1. Statistical: Identifying outlier fields and records using the values of mean, standard deviation, range, etc., based on Chebyshev's theorem [13, 18], considering the confidence intervals for each field [52]. Chebyshev's theorem is as follows. Let X be a random variable with finite expected value μ and non-zero variance δ^2. Then for any real number $k > 0, \Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$. Using Chebyshev's theorem, outlier values for particular fields are identified based on automatically computed statistics. For each field the average and the standard deviation are utilized and based on Chebyshevs theorem those records that have values in a given field outside a number of standard deviations from the mean are identified.

2. Clustering: Identify outlier records using clustering based on Euclidian (or other) distance. Existing clustering algorithms provide little support for identifying outliers [55, 72, 105]. However, in some cases clustering the entire record space can reveal outliers that are not identified at the field level inspection (chapter 11 in [52]). The main drawback of this method is computational time. Standard clustering algorithms have high computational complexity. For large record spaces and large number of records, the run time of the clustering algorithms is prohibitive.

3. Pattern-based: Identify outlier fields and records that do not conform to existing patterns in the data. Combined techniques (partitioning and classification) are used to identify patterns that apply to most records [53]. A pattern is defined by a group of records that have similar characteristics (”behavior”) for p% of the fields in the data set, where p is a user-defined value (usually above 90).

4. Association rules: Association rules with high confidence and support define a different kind of pattern. As before, records that do not follow these rules

are considered outliers. The power of association rules is that they can deal with data of different types.

The term association rule was first introduced by Agrawal et al. [3] in the context of market basket analysis. In this analysis, the data set is defined as the basket data $B = \{b_1, b_2, ..., b_n\}$, where each basket $b_i \in I$ is a collection of items, and where $I = \{i_1, i_2, ..., i_k\}$ is a set of k elements. An association rule in the database B is defined as follows.

$i_1 \Rightarrow i_2$ is an association rule [3] if:

1. i_1 and i_2 occur together in at least $s\%$ of the n baskets, then s is called the support of the rule;

2. and, of all baskets containing i_1, at least $c\%$ contain i_2, then c is called the confidence of the rule.

This definition extends easily to $A \Rightarrow B$, where A and B are disjoint sets of items instead of single items. In general A is referred to as the antecedent (or left-hand side) of the rule, and B as the consequence (or right-hand side) of the rule. In real-life cases and spoken language terms, an association rule is phrased as: “50% of people who buy diapers, also buy beer, and 20% of all buyers buy diapers.” In this case, diapers and beer are the items, the confidence of the rule is 50%, and the support of the rule is 10%. Association rules of this type are also referred to in the literature as *classical* or *boolean* association rules. In practice, the information in many, if not most, databases is not limited to categorical attributes, but also contains much quantitative data. Unfortunately, the definition of categorical association rules does not translate directly to the case of quantitative attributes. It is therefore necessary to provide a definition of association rules for the case of a database containing quantitative attributes. Srikant and Agrawal [88] extended the categorical definition to include quantitative data. The basis for their definition is to build categorical events from the quantitative data by considering intervals of the numeric values. Thus, each basic event is either a categorical item or a range of numerical values. Such rules are called quantitative association rules [88]. A more formal definition is given there, and confidence and support of the rule are slightly redefined. An example of such a rule is “People who spent on bread between \$3-\$5, and on milk at the same time between \$1-\$2, usually spend between \$1.5-\$2 on butter in the same transaction.” A stronger set of rules is defined in [56] as ratio-rules. A rule under this framework is expressed in the form: “Customers typically spend 1: 2: 5 dollars on bread: milk: butter”. This time the strength of the rule allows multiple applications, including data cleansing and outlier detection. However, the paper does not exploit this idea. It is only mentioned that the power of ratio-rules to reconstruct data could support the data cleansing process. Eigen system analysis is used to find these rules and induces the strength of the rules as well as a computational overhead. A series of generalizations of quantitative association rules are defined in [10]. Ordinal association rules were defined by the authors [65] and used to find rules that gave more information (e.g., ordinal relationships between data elements). A further generalization is made in [77] where a general form of rules are considered: *body* ⇒ *head*, where body is a conjunction of atomic conditions of the form *attribute op value*, and *head* is a single atomic condition of the form *attribute op value*, where $op \in \{\leq, =, \geq\}$. These generalized association rules yield a special type of patterns, so this method is, in general, similar with the pattern based method.

2.3 Data Quality on the Semantic Web

This section contains two parts. The first part is about how to annotate the quality information on existing Semantic Web data. The other part is about how to evaluate the quality of Semantic Web data.

2.3.1 Quality Annotation

The basic principle of quality annotation is that the data contains explicit info of the validity of individual pieces of data, or triples in the case of RDF. This section details various way to add annotations of validity directly to the triple themselves, such annotations are hereby referred to as “quality annotations.” For the sake

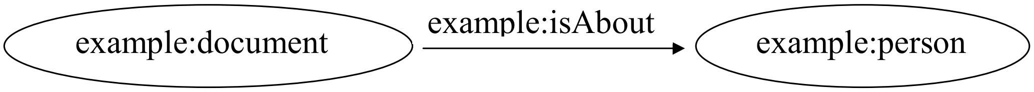


Figure 2.3: A basic example triple.

of clarity, they all contain a way of marking the triple portrayed in Fig. 2.3 as imperfect. It should be noted though that because of the methods different, so are the semantic implications of them. As such, all of the figures to be presented are not exactly semantically equivalent to each other.

Reification is a feature in RDF that, even though it is fundamentally represented with triples, is a deviation from the basic graph/triple structure. Reification means the process of reifying an RDF statement (triple) by giving it an unique identifier as a whole: this enables it to be used as a part of another triple, either the subject or the object. If used as the subject, instead of the regular (subject - predicate - object) structure it would be ((subject - predicate - object) - predicate - object). This allows native support for more complicated semantic structures such as "Weather was sunny at noon", which would otherwise be non-trivial to implement with triples. Reification can be used for quality annotations, by marking triple-specific metadata to support or doubt the validity of triples [26]. A simple method of quality annotation markup with reification is shown in Fig. 2.4. Statements have been marked as uncertain by linking them to a general "uncertain resource" via an object property. This method is general in the sense that the properties can be arbitrary and there can be many of them. For example, we could easily give a numerical confidence value for the uncertainty with a literal-valued triple. However, reification has the problem on the complexity of this method, because reification partly breaks the general triple structure of (subject-predicate-object) statements model in RDF. This means that reification should be separately taken into account in any application that uses the data. The most common use cases for semantic web data, such as recommendation systems and automatic data enrichment, are considerably harder to implement and computationally more expensive with data that contains reified statements [42, 11].

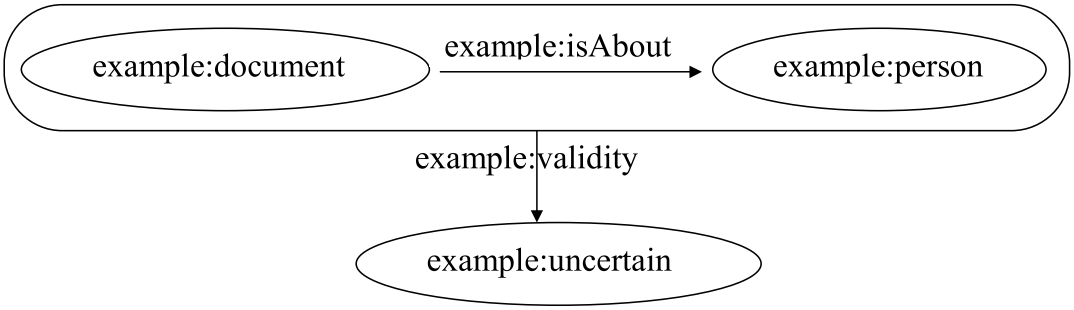


Figure 2.4: Data quality annotation with reification.

One method to achieve the wanted goal is to use named graphs for storing extra information about a triple [97]. Basically this means that instead of triples, the data store utilizes quads, where the subject-predicate-object combination additionally has a named graph (URI) attached. Once the triples are linked to named graphs, we can store information about the graphs: for example, their origin, method of creation and authors. This kind of extra information is usually called provenance, and the method is widely used on the Semantic Web by many large organizations. The method is somewhat similar to the reification method described above, but is simpler and more elegant as it clutters the data less and is more intuitive and transparent. However, it is more suited to provide provenance than direct quality information since that is its original purpose. If we use it for quality information instead, like in Fig. 2.5, we must tag the whole graph with the same quality information, or alternatively divide the graph into a multitude of subgraphs. Having the ability to annotate single triples and resources is useful, but dividing the graph into small subgraphs is counterproductive if we also want to annotate the provenance of a graph as a whole. This is a problem as both quality and provenance information should be allowed to coexist in the same RDF dataset. An alternative is to simply track the quality of whole datasets instead of smaller units such as triples or resources [44], but this would be a much less broad and general approach.

For certain applications, linking uncertain data into an existing knowledge base is enough to solve the data quality annotation problem. If we have a knowledge base we know we can trust, and some unknown data whose quality we want to annotate,

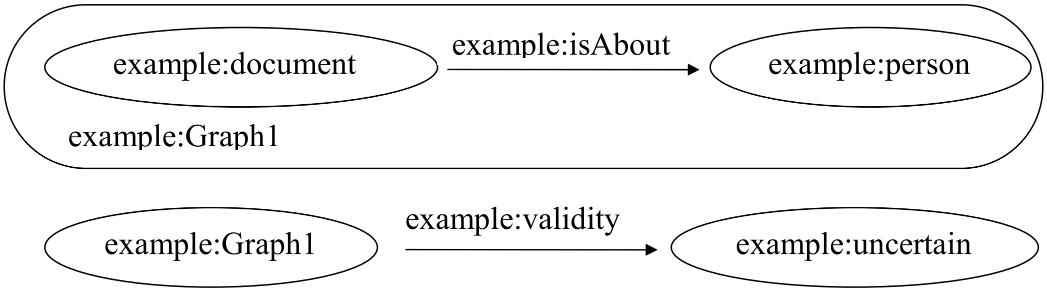


Figure 2.5: Quality annotation with named graphs in quads.

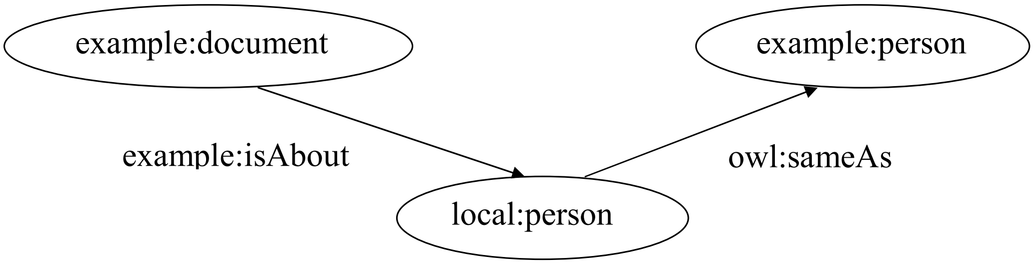


Figure 2.6: Data mapping using local instances.

it's sometimes enough to create a mapping between these datasets instead of actually figuring out confidence values and other metadata. Fig. 2.6 illustrates this process: we create a local instance of every possibly interesting resource, and link these to a pre-existing knowledge base using, for example, the property $owl : sameAs$ if we consider them to be exactly the same, or some other relation. In the case of reference ontologies and classes, this relation is often $rdfs : subClassOf$, as it is a very strong thing to say that two classes as exactly the same. This marks them as being semantically similar and links them to the whole knowledge base, whereas local instances deemed as invalid are left to be orphans or semantic "dead ends", without links to meaningful resources. Due to the nature of this method, it has limited use. Primarily it can be used in the context of automatic annotation and converting legacy data over to the semantic web [94]. In this application, everything is about linking old data to new data, essentially the equivalence of resources.

2.3.2 Semantic Web Data Evaluation

Most existing work on semantic web data evaluation focuses on ontology evaluation which emphasizes syntax validation and logical consistency analysis. Sabou et al. [82] predict the correctness of ontology axioms using the corroboration of the axioms (both explicit and entailed) from other ontologies, but the corroboration requires the statements to be exactly the same as the original axiom, which is limited and unreliable considering unreliable data existing on the Web.

The first group of work on instance data evaluation focuses on if the data usages are explicitly consistent with the syntax of the ontologies. Before consuming semantic web data, many applications and users deploy automated tools to find problems early. Typically users will deploy a syntax checker, such as the W3C RDF Validator (`http://www.w3.org/RDF/Validator/`). It checks whether an RDF document conforms to RDF/XML syntax. The validation is backed by the ARP parser of Jena [68]. This tool contributes to a critical step in instance data evaluation - parsing RDF data to RDF triples. The second type of errors is logical inconsistency which may be checked by deploying reasoner tools such as Pellet. These tools typically provide techniques aimed at finding provable (rather than possible) issues. For instance, a logic problem could be that a URI type of class A is used as the subject of a property whose domain is declared a class type B which is declared as disjoint with A.

The other category of works focuses on checking if the data uses the vocabularies in the ontologies appropriately on semantics. For instance, a semantic error could be that the topic of a paper published on Semantic Web conference is chemistry. Obviously all the syntax here is correct, so any syntax based or logic based mechanisms will not work and can not give any suggestions. Tao et al. [91] captured some symptoms of potential data quality issues by using a conjunctive combination of the presence and absence of certain triple patterns in certain dataset. They listed three sub-categories:

1. Potential issues related to an individual type. In this category of potential issues, there are three specific symptoms.

(a) First, we are given an individual i, and all of its types $\{t_1, t_2, ...\}$ in the deductive closure (DC) of instance data, including the declared types and inferred types. Then, (i) a logical inconsistency will occur if there is a declared $owl : disjointWith$ relation between a pair of t_i and t_j; (ii) no issue will occur if there is an rdfs:subClassOf relation between any pair of t_i and t_j; (iii) A potential issue will occur otherwise. The reason of the potential issues here is that no subset or disjoint relation can be found between two expected types of an individual: one comes from the types declared in DC and ontologies and the other is inferred from properties rdfs:domain, rdfs:range and owl:allValuesFrom in DC.

(b) Second, given an individual i, assume its types declared in the instance data are $\{td_1, td_2, ...\}$, it may also have some types declared in instance data or the referenced ontologies $\{tdo_1, tdo_2, ...\}$, if any two types td_i and tdo_j from these two groups of types having a subset relation (not the same), a potential issue will occur because this instance is declared as both an instance of a class and its super class at the same time. The reason of the potential issues here is that an individual has redundant individual types.

(c) Third, we are given an individual i, and its declared types $\{t_1, t_2, ...\}$ in instance data and referenced ontologies. If there is a type t_i which has some sub-classes $\{t_{i1}, t_{i2}, ...\}$ in instance data including inferred data, but i is not known to be an instance of any of the subclasses, a potential issue will occur. The explanation of this symptom is as follows. An individual is stated to be an instance of a class which is a non-leaf node in the class hierarchy. Usually this is because the instance data author forgets to provide more specific type after assigning the individual a general type, or the author does not know the instance's specific type. The reason for the potential issues here is that the instance has a non-specific individual type.

2. Potential issues related to a property value. In this category of potential issues,

there are two possible symptoms.

   (a) We are given an individual i that has type c, let m be the number of triples (i, p, x) in the data, where p is a property. If there is a cardinality restriction in the ontology, requiring all instances of c to have at least n values for property p, an issue will occur if $m < n$. The reason that causes the potential issues here is the violation of the owl:cardinality or owl:minCardinality restriction. Apparently, the authors are making the closed world assumption. Otherwise, they are saying there are some triples for which the values are unknown.

   (b) The next symptom is that, similarly, an issue would occur if there are more (i, p, x) triples in the data than expected by referenced ontologies. The reason that causes the potential issues here is the violation of the owl:cardinality or owl:maxCardinality restriction. In this case, the authors are assuming there are no owl:sameAs statements (explicit or implicit) between the x values.

3. Other application-specific potential issues. These expectations require a flexible and extensible evaluation approach, so that users can customize their own evaluation criteria to check issues beyond syntax, logical consistency, and those frequently encountered potential issues. For example, some applications may require instances of some classes to always be named, i.e. they must be a URI instead of a blank RDF node. In other situations, users may require all instance descriptions to have an annotation property rdfs:label.

For each of the above symptoms, the authors give a SPARQL query to find such data that satisfies the conditions. However, as the authors pointed out, the list of issues is far from exhaustive.

Another type of work on evaluating instance data is based on the context where the data is used, for example in the context of the results for a query or within the data that uses a group of similar predicates. To rank query results, Stojanovic et al. [90] pointed out that there are two differences among returned relation instances for answers to a query. They are the specificity of the instantiation of the ontology and the inference process. The specificity is on the basis of the number of interpretations of the given relation instance. A relation type is defined by the relation that is instantiated in the relation instance. The specificity of a relation instance can be interpreted as the measure of its relevance for the user's query. When the specificity of a relation instance is higher, then the relevance is higher as well. Anyanwu et al. [7] classified the ranking criteria for complex relationship search results into semantic and statistical metrics. One of the semantic metrics is a query context by allowing users to interact with a graphical visualization of the ontology to specify the query context. However most common users have difficulties in understanding an RDF graph and each term in it, since even knowledge representation professionals frequently disagree on the meaning of the same entity or expression in different ontologies. Another semantic metric used by Anyanwu et al. is trust. When computing trust weights of a semantic association, the strength of an association is only as strong as its weakest link. However trust values on every link were empirically assigned in their work. The statistical metrics are the measures of rarity and popularity of each component in the semantic association. A semantic association represents semantic similarity between paths connecting different resources in an RDF model. I observe that the metrics in these two works are both rankings of resources from a global perspective rather than from each queries' specific perspectives. Franz et al. [35] provided an approach via tensor decomposition to classify predicates into pre-defined number of groups and compute authoritative scores for resources with respect to every group. This is a finer grained authoritative measurement than previous works in this group. However the ranking is still focusing on resources as others instead of to give rankings for a triple, especially when a query consists of triples across multiple domains.

To the best of my knowledge, the above discussions are complete lists of previous works that are closely related to this thesis. Ideally, there are some existing works for comparison with my systems on performance. However, it can be noted in above discussions that there is no existing work focus on the exactly same topic as ours, i.e. automatically detecting abnormal Semantic Web instance data. First, some works related to the Semantic Web only focus on ontologies or uses some manually created ad-hoc rules for detecting potential quality issues. So the scope that systems focus are different. Second, the works detecting quality issues in other types of data, e.g. databases, can not be easily adapted to deal with Semantic Web data. Further it is also hard to convert RDF data to relational databases. Thus these traditional data cleansing algorithms cannot be used for comparison. Third, most outlier detection approaches focuses only on numerical data by checking if some values are markedly away from the majority. However my systems are designed to deal with object property values and all types of datatype property values (both strings and numbers). Thus it would be unfair to compare them. Therefore, in this thesis, I did not compare the performance of my systems with other existing systems on detecting abnormal Semantic Web data. But to evaluate each part of my systems, I designed and conducted various, extensive experiments to check the usefulness and necessity of each (see each chapter for algorithms).

Chapter 3

The Problem of Detecting Abnormal Semantic Web Data

To better define our problem of detecting abnormal Semantic Web data, I will introduce its definition by comparing with outlier detection in relational databases. Specifically, I begin with a discussion of the concept of ouliers and follow with a review of the general methods used in outlier detection. Next I give the definition of abnormal Semantic Web data before I describe the design considerations of my approaches for detecting abnormal Semantic Web data.

3.1 Outliers

Despite the importance of data collection and analysis, data quality remains a pervasive and thorny problem in almost every large organization. The presence of incorrect or inconsistent data can significantly distort the results of analyses, often negating the potential benefits of information-driven approaches. As a result, there has been a variety of research over the last decades on various aspects of data cleaning: computational procedures to automatically or semi-automatically identify - and, when possible, correct - errors in large data sets. Outlier detection is an important area of such research. Although outliers are often considered as an error or noise, they may carry important information. Detected outliers are candidates for aberrant data that may otherwise adversely lead to model misspecification, biased parameter estimation and incorrect results. It is therefore important to identify them prior to modeling and analysis.

An exact definition of an outlier often depends on hidden assumptions regarding the data structure and the applied detection method. Yet, some definitions are general enough to cope with various types of data and methods. Hawkins [46] defines an outlier as “an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism”. Barnet and Lewis [13] indicate that “an outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs, similarly, Johnson [52] defines an outlier as an observation in a data set which appears to be inconsistent with the remainder of that set of data”. Wainer [95] also introduced the concept of the “fringelier,” referring to “unusual events which occur more often than seldom.” The definition by Grubbs [39]: “An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs”, is more often referenced.

Outliers can arise from several different mechanisms or causes. Anscombe [6] sorts outliers into two major categories: those arising from errors in the data, and those arising from the inherent variability of the data. Not all outliers are illegitimate contaminants, and not all illegitimate observations show up as outliers [13]. It is therefore important to consider the range of causes that may be responsible for outliers in a given data set. Osborne and Overbay [76] summarized them as follows.

1. Outliers from data errors. Outliers are often caused by human error, such as errors in data collection, recording, or entry. The example about radiation tests conducted by Transportation Security Administration that I gave in the beginning of the introduction chapter is one of these errors caused by human entry.

2. Outliers from intentional or motivated mis-reporting. Sometimes participants

intentionally report incorrect data to experimenters or surveyers. A participant may make a conscious effort to sabotage the research [48], or may be acting from other motives.

3. Outliers from sampling error. Another cause of outliers or fringeliers is sampling. It is possible that a few members of a sample were inadvertently drawn from a different population than the rest of the sample.

4. Outliers from standardization failure. Outliers can be caused by research methodology, particularly if something anomalous happened during a particular subject's experience, e.g. observations in a classroom the day before a big holiday recess.

5. Outliers from faulty distributional assumptions. Incorrect assumptions about the distribution of the data can also lead to the presence of suspected outliers [50]. For example scores on classroom tests where students are well-prepared may be a flat distribution.

6. Outliers as legitimate cases sampled from the correct population. It is possible that an outlier can come from the population being sampled legitimately through random chance. It is important to note that sample size plays a role in the probability of outlying values. Within a normally distributed population, it is more probable that a given data point will be drawn from the most densely concentrated area of the distribution, rather than one of the tails [30, 83].

As listed above, the most important source of outliers is from data errors which is also the most important motivation to detect such outliers. Thus it is also critical to know the causes of original data errors, especially in a database. Maydanchik [67] summarized them and here I list some main causes of them in the following.

1. Initial data conversion. Databases rarely begin their life empty. During the data conversion it is the data structure that is usually the center of attention. The data is mapped between old and new databases. However, since the business rule layers of the source and destination systems are very different,

this approach inevitably fails. The converted data, even if technically correct, is often inaccurate for all practical purposes.

2. System consolidations. Database consolidations are the most common occurrence in the information technology landscape. They take place regularly when old systems are phased out or combined. And, of course, they always follow company mergers and acquisitions. The data is often merged into an existing non-empty database, whose structure can be changed little or none whatsoever. However, often the new data simply does not fit!

3. Manual data entry. Despite high automation, much data is (and will always be!) typed into the databases by people through various forms and interfaces. To err, after all, is human!

4. Batch feeds. Batch feeds are large regular data exchange interfaces between systems. The source system that originates the batch feed is subject to frequent structural changes, updates, and upgrades. Testing the impact of these changes on the data feeds to multiple independent downstream databases is a difficult and often impractical step.

5. Real-time interfaces. The basic problem is that data is propagated too fast. There is little time to verify that the data is accurate. Further, the data comes in small packets, each taken completely out of context. A packet of data in itself may look innocent, but the data in it may be totally erroneous.

6. Data purging. When data is purged, there is always a risk that some relevant data is purged by accident.

3.2 Outlier Detection

Outlier detection methods have been suggested for numerous applications. Besides data cleansing, they also include credit card fraud detection, clinical trials, voting irregularity analysis, network intrusion, severe weather prediction, geographic information systems, athlete performance analysis, and other data-mining tasks. Typically, these methods focus on quantitative data.

Quantitative data is integers or floating point numbers that measure quantities of interest. Quantitative data may consist of simple sets of numbers, or complex arrays of data in multiple dimensions, sometimes captured over time in time series. Quantitative data is typically based in some unit of measure, which needs to be uniform across the data for analysis to be meaningful; unit conversion (especially for volatile units like currencies) can often be a challenge. Statistical methods for outlier detection are the foundation of data cleaning techniques in this domain: they try to identify readings that are in some sense "far" from what one would expect based on the rest of the data. In recent years, this area has expanded into the more recent field of data mining, which emerged in part to develop statistical methods that are efficient on very large data sets.

3.2.1 Univariate

The simplest case to consider - and one of the most useful - is to analyze the set of values that appear in a single column of a database table. Many sources of dirty quantitative data are discoverable by examining one column at a time, including common cases of mistyping and the use of extreme default values on numeric columns. This single-attribute, or univariate, case provides an opportunity to introduce basic statistical concepts in a relatively intuitive setting.

It can be difficult to define the notion of an outlier crisply. Given a set of values, most data analysts have an intuitive sense of when some of the values are "far enough" from "average" that they deserve extra scrutiny. There are various ways to make this notion concrete, which rest on defining specific metrics for the center of the set of values (what is "average") and the dispersion of the set (which determines what is "far" from average, in a relative sense).

The center, or core, of a set of values is some "typical" value that may or may not appear in the set. The dispersion, or spread, of values around the center gives a sense of what kinds of deviation from the center are common. The most familiar metric of dispersion is the standard deviation, or the variance, which is equal to the standard deviation squared.

The “center/dispersion” intuition about outliers defines one of the most familiar ideas in statistics: the normal distribution, sometimes called a Gaussian distribution, and familiarly known as the bell curve. For example, a typical definition of an outlier is any value that is more than 2 standard deviations from the mean.

3.2.2 Multivariate

As an example of multivariate case, consider a table of economic indicators for various countries, which has a column for average household income, and another column for average household expenditures. In general, incomes across countries may range very widely, as would expenditures. However, one would expect that income and expenditures are positively correlated: the higher the income in a country, the higher the expenditures. So a row for a country with a low per-capita average income but a high per-capita average expenditure is very likely an error, even though the numbers taken individually may be well within the normal range. Multivariate outlier detection can flag these kinds of outliers as suspicious.

Multivariate techniques are analogous in some ways to ideas I introduced in the univariate case, but the combinations of variables make things both more complicated to define and interpret, and more time-consuming to compute. Most approaches extend the basic measures of center and dispersion to the multivariate setting.

The problem with this extension is the use of a single number for the dispersion around the center. That single number can only define distributions that are symmetric about the center, which is not enough to capture x/y correlations like the one I introduced between income and expenditures. To capture these correlations more generally, a multivariate normal distribution is defined by a multivariate mean and a two-dimensional covariance matrix, which can be thought of as a generalization of the univariate variance to multiple variables. Recall that the variance of a single variable is the standard deviation squared. The covariance matrix essentially captures the variance of each variable individually, and the correlation (covariance) between pairs of variables.

3.3 Abnormal Semantic Web data

A major share of Semantic Web data originates from existing relational databases and is lifted by mapping database schema elements to Web ontologies. For example GoodRelations [47] is the ontology for publishing the details of products and services in a way friendly to search engines, mobile applications, and browser extensions. Because of the process of lifting existing data sources to the RDF data model and Web ontologies, existing data quality problems from the original representation would be usually replicated. So on the one hand, all the reasons that cause the data quality issues and make them outliers discussed in the previous section are also possible reasons for quality issues of the data on the Semantic Web. On the other hand, the transformation of data from original forms to the Semantic Web form also can bring in quality issues due to inappropriate extraction, mapping, creation, etc. A typical semantic knowledge extraction from text has the following steps. First instances or named entities are identified and extracted. Then they are mapped to the object identifier in the Semantic Web data. Meanwhile the relation between this entity with other entities mentioned in the text are also mapped to predicate in the ontologies. Finally the triples consisting of these entities and relations are created. While sophisticated conversion scripts and middleware components can filter out some of the problems, the negative impact of data quality issues will grow on the Web of Data, because the data will be used in more applications and in more unique contexts. The amount and impact of any problems will increase accordingly.

Since the previous definitions of outliers have been so many and general, I try to give a specific definition from the Semantic Web unique perspective. Meanwhile the definition is also general enough to cover almost all major types of quality issues on the Semantic Web. Further the definition ideally needs to give easy guidance of quality control. The triples on the Semantic Web are the atomic information unit. Different from free text or other semi-structured documents, recall a statement consists of three parts: subject, predicate and object. In other words it describes a relationship between two resources or a resource and a literal. For a piece of data consisting of these three parts, there are eight possible situations of the errors on the three positions, from the one that none of the three is erroneous to the one that all the three are erroneous. So among them, seven combinations would cause the triple be erroneous as a whole and they are listed in Table 3.1. Then we can generalize all these possible errors that a triple can have into three types (shown in Table 3.1).

1. The subject or object is incorrect, i.e. it is not supposed to be put into the statement. For example the triple $< U.S., residesIn, South\ America >$ has an incorrect object (since U.S. is located in North America) and the triple $< James\, Cameron, directorOf, Star\ Wars >$ has an incorrect subject (since George Lucas is the director of Star Wars). However, unless we know the data creator's original intention, both types of errors are interchangeable. For instance, the above two examples can be claimed to have either type of error. To make the above both errors more general, we can categorize them as that the pair of subject and object has no significant relation with respect to the predicate defined in the ontology vocabulary. In this definition, the above two examples have the same error that the subject and object do not have the relationship indicated by the predicate used. We emphasize that it is with respect to the ontologies, because any two objects can be put together and be linked by an arbitrary relation, e.g. hasRelation. But the data is described in accordance to certain ontology vocabulary and so using the concepts other than those defined in that vocabulary would have no meaning and cause confusion. Thus the significant relation that we focus on is tied to the ontologies and the data set where the statement is in.

2. The actual relationship between the subject and the object is not consistent with the predicate used in this triple, i.e. the subject and object should be related by another predicate. For example, $< Bill\, Clinton, motherOf, Chelsea\ Clinton >$ has an incorrect predicate usage.

Table 3.1: All possible errors that a triple can have.

subject	predicate	object	type
×			1
	×		2
		×	1
×	×		3
×		×	3
	×	×	3
×	×	×	3

3. The combination of the previous two types of errors, i.e. more than one elements among the subject, the predicate and the subject are incorrectly used in this statement. These errors can be detected if using the solutions for the previous two types of errors.

Having the above list of all possible errors that could occur in Semantic Web data, then I discuss abnormal Semantic Web data in corresponding to these errors as follows.

1. In the above, I have generalized the error occurred in the subject or the object field as a type of error that there is no relationship between the subject and the object. Then the abnormal Semantic Web data that could be an heuristic of this type of error would be this symptom. Given a triple in the data set, most of pairs of objects in the same data set that are similar to the pair of objects in this triple have no relationship.

2. In the above, I have generalized the error occurred in the predicate field as a type of error that there is an incorrect relationship between the subject and the object. Then the abnormal Semantic Web data that could be an heuristic of this type of error would be this symptom. Give a triple in the data set, most of pairs of objects in the same data set that are similar to the pair of objects in this triple have a different relationship.

3. All the other type of errors could show the combination of the above two symptoms.

In sum of these three symptoms, an abnormal Semantic Web triple, is one that has an abnormal relation that appears to deviate markedly from other pairs of similar objects in the data set which it occurs. The deviation could be because either there is no relation between other similar pairs of objects or there is a different relation is often used between other similar pairs of objects. This definition gives clear guidance on the process to detect such deviations, i.e. checking the relationship between the subject and the object.

3.4 Design Considerations of a Practical System

The key theme of the work is to develop a set of data structures and algorithms that will give suggestions on the correctness of the Semantic Web instance data. Since the Semantic Web represents many points of view, there is no objective measure of correctness for all Semantic Web data. Therefore, we focus on the detection of abnormal triples, i.e., triples that violate certain rules learned from most of the data or from a set of verified data. This in turn is used as a heuristic of a potential data quality problem. We recognize that not all abnormal data is incorrect (in fact, in some scenarios the abnormal data may be the most interesting data) and thus leave it up to the application to determine how to use the heuristic. This explanation of abnormality is exactly the same as the core idea of outlier detection. Thus some approaches for outlier detection for data cleansing can inspire my solutions, such as univariate/multivariate correlation discussed above, association rules and pattern-based (both are discussed in section 2.2.3), etc. For example, the approach to be designed can connect multiple variables/columns, i.e. multiple properties in Semantic Web data, and discover patterns across them; the rules to be discovered can also comprise an antecedent (or left-hand side) of the rule, and the consequence right-hand side) of the rule, as the association rules. All these methods can be generalized as discovering characteristics of data first and then apply the characteristics to detecting anomalous data that does not follow the characteristics. Starting from this general methodology, we need to adapt it to the Semantic Web data environment in order to detect abnormal Semantic Web data defined above. Because ontologies serve as the formal vocabulary for Semantic Web data, the correctness in this work is with respect to certain ontologies that the data conform to, i.e. if the semantics are correctly conveyed by descriptions of objects using the vocabulary from the ontologies.

Compared to the database outlier detection, there are a number of challenges in apply similar approaches to Semantic Web data. First, typical relational database is organized into relational tables. A table consists of a group of similar records that describing the same set of properties for these objects. However, RDF model is more flexible and there is no table for grouping objects. Thus it is a challenge to find a natural group of objects and properties for them in order to discover potential common characteristics. Second, the schema in database is comparable to the ontologies that a Semantic Web data set conforms to. However the ontologies provide more deeper semantics and logics that can be used to infer new knowledge. Thus it is a challenge to take into account the advantages provided by the ontologies. Third, the open world assumption is more often applied to Semantic Web data compared to databases. Thus it is a challenge to better interpret the difference between existing data and non-existing data for learning true potential characteristics underlying the data set. Fourth, the errors that could occur in Semantic Web data can be caused by more reasons and in more forms. For example the faulty inference based on incorrect sameAs triples in Semantic Web data could produce new incorrect triples. Fifth, traditional database outlier detection mainly focus on numeric values. The essential idea of using statistics, data mining techniques in database outlier detection is still valid to detecting abnormal Semantic Web data. However, it is a challenge to find a general method to detect all types of values in Semantic Web data.

To give suggestions on correctness of RDF data, the two main technical goals of this work are (1) how to find potential probabilistic rules underlying RDF data that conform to certain ontologies and (2) how the system can use it in different scenarios. To test if the system succeeds in the goal, we can input Semantic Web data to let the system apply learned probabilistic rules and then check the difference between inference results and the triples' actual state (ground truth). After the probabilistic rules are validated, the inference results of the system can be used as report of a degree of abnormality. Specifically, for different situations of real world Semantic Web data, I designed the following four algorithms.

1. In the first situation, the system relies on a good training data set and a relatively strong assumptions - the closed world assumption. The most important two questions I want to answer are as follows. First, given a piece of data, can its contextual data serve as evidence that it is "normal"? Second, are there indeed some useful patterns existing in the contexts. Because the essential idea of evaluation is based on the contextual information of data, a good training data set requires two characteristics. (1) It needs to be a well described data set, i.e. instances on the RDF graph are well connected through using predicates (edges) and reusing object values (nodes); (2) the data is verified or assumed to be generally correct. The strong assumption is that the closed world assumption applies to the data, i.e. all relevant statements not in the data are false. This assumption is also a typical categorized situation in traditional data cleansing (I discussed this data quality dimensions in section 2.2.1). In this case, we categorize possible errors that a triple can have into two types. For each type of error, the system uses classification techniques to differentiate the data based on the features learned from the referenced data. The features include numeric metrics measuring the credibility of the relation between the subject, the object and patterns measuring the type of relation that is the most likely, etc.

2. In the second situation, we want to weaken the system requirements on training data set and assumptions. In other words, the closed world assumption does not apply to the data set, i.e. we cannot assume that the relevant statements not in the data are false. As previous research on data cleansing under the open world assumption has shown [9], a value can be missing either because it exists but is unknown, or because it does not exist at all, or because it may exist but it is not actually known whether it exists or not. Thus under this situation, I devise a learning model that tries to avoid interpreting the data

into binary 0/1 scheme, i.e. it encodes each positive/existing sample as 1 and interprets the remaining/non-existing data as 0. In addition, I try to improve features used in classifiers in previous work by making them concise enough so that the system can become an more integrated general classifier for all types of possible errors that a object property triple can have.

3. In the previous two systems, I recognized that in reality, some data sets cannot be treated as generally correct for learning. And to verify the fact that a data set is generally correct, it may require lots of human effort to manually check them and thus is impractical. Thus the third case that I want the system to deal with is where there is no generally correct data set for learning in advance. The previously described scenarios focus on detecting anomalies in new data after training on normal (or clean) data. However, there are many cases where we cannot find such a clean data for training and we need to detect anomalies directly in a noisy data set. Thus a new algorithm needs to be designed for detecting abnormal Semantic Web data in a data set that contains a large number of normal elements and a significant portion of abnormal data as well. The system can try to discover patterns similar to previous systems. But, importantly, if the data set that patterns are extracted from contains a significant number of errors (i.e. it could have significant portion of unreliable or noisy data), the patterns would be not very reliable and should be better weighted according to the reliability. Then I should have a mechanism to automatically adjust them, e.g. iteratively, based upon other aspects, e.g. consistency among patterns. Through this process, the patterns are expected to be gradually differentiated and so the data can be differentiated according to the support of patterns.

4. The patterns discovered and used by my previous works are mainly based on the explicit connections between data, i.e. the reused values. Thus we need an iterative approach to gradually differentiate data by majority voting, because the patterns are limited and not strong enough, But for data sets where few statements are explicitly connected through reusing the same values

and, more generally, for most data using datatype values, the patterns that can be found can be even more limited. Therefore we need to find more patterns and might only focus on strong patterns for efficiency both on searching and applying these rules. Starting to think about this problem, I notice that the patterns discovered by my previous systems are close to the concept of data dependency, especially functional dependencies. As I introduced in section 2.2.1, some dependencies have been put into practice for data quality research before. Therefore, a good intended extension on data dependency should be a promising approach for the Semantic Web data too. The extension is expected to cover most of the patterns in previous system as well as some new patterns that could be unique in RDF data. Because RDF data can be viewed as a graph data model, I also can extend functional dependency into value-clustered graph functional dependency. They are devised to capture more implicit correlations among values in different statements. If I can abstract them into probabilistic integrity constraints, they would greatly improve the system capability on detecting abnormal Semantic Web data.

The following chapters will discuss each of these algorithms in details respectively.

Chapter 4

Data Correctness under the Closed World Assumption

This chapter introduces my first effort towards high quality Semantic Web data. Because it is the first step, I want to make sure that there are some features in data that can be used to automatically categorize the quality of different portions. Therefore I start with a relatively simple situation with relatively strong assumptions. I refer to this scenario as detecting abnormal Semantic Web data under closed world assumption [100]. The closed world assumption is the presumption that what is not currently known to be true, is false. Specifically, on the Semantic Web data domain, the closed world assumption means that for every object in the domain, the values of every applicable property for this object are given in the data. Dividing real world situations according to this dimension is not new in data quality research, it is also one of important aspects for traditional data quality dimensions (discussed in section 2.2.1). As I have discussed in the previous sections, the main technical problems that a solution tries to solve are: 1) how to find common characteristics underlying the data; 2) how to use these characteristics to detect the data that does not follow them. The solution to be introduced in this chapter is based on discovering characteristics from a training data set first and then comparing between the training dataset and the data being investigated. The training dataset has to satisfy two requirements. First it is generally correct. This requirement means that the data set could have few errors and no consistent repeated errors, i.e. no systematic errors. Although it is hard to quantify the percentage of errors that is allowed, based on our experiments (will be introduced in subsequent subsections), when the erroneous data is less than 5%, it usually would not affect the general patterns in the data set. The second requirement to the data set is that it is comprehensively described with respect to the vocabulary of the ontologies, i.e. the data gives much contextual information for each triple. I emphasize that the comprehensiveness is important here and it is an approximate interpretation of the closed world assumption: given the ontology vocabularies, most of objects in the domain which the dataset describes have rich usages of all available concepts and properties.

4.1 Approach

The Semantic Web has an advantage over the databases in that it supports semantics specified entailment. Semantic Web data follows the logics in the ontologies that are used as vocabulary to describe them. If we take entailment as a corroboration, data that is entailed by existing knowledge is more likely to be correct than which is not entailed. However in real world datasets it is not surprising that the ontologies can not be found, ontology definitions are too simple, or data is created without much deep logic consideration. All these real world situations would make invalid the basic conditions for logic inference. Thus since this kind of corroboration is too strong and so limited, I need to find some weaker corroboration. The problem of detecting abnormal data can naturally be viewed as a classification problem: classifying data into two general categories, either normal or abnormal. Compared to formal logic inference, classification needs weaker conditions and is more general. It is also a typical perspective often used in traditional statistical data cleansing, especially the pattern-based methods for error detection (discussed in section 2.2.3). This type of methods combine techniques, such as partition, classification, etc., to identify patterns that apply to most data. To build a classifier, features need to be identified that can be used to build the classifier at first. As defined in section 3.3, the errors in Semantic Web data can be generalized as incorrect relational descriptions between the subject and object in triples. Thus, a candidate feature for a relational classifier is the context for the subject/object pair that consists of various direct/indirect relations between them.

There are also several observations about using contextual information to assess correctness. First, a Semantic Web dataset normally is about a certain topic, on a certain domain and for a certain usage. Second, the number of ontologies used in the dataset is commonly limited to a few, i.e. concepts and properties for this domain are limited. Thus the data in it are probably described and used in certain common ways. If a piece of data contains some errors caused by accidental input or misunderstanding of ontologies, it would be expected to show certain abnormal contextual evidence compared to the majority of the dataset that are similar to it. Thus there could exist some patterns that I am looking for. If potential patterns underlying the whole dataset can be discovered, the patterns can be used to detect some abnormal data or give suggestions for the data to evaluated. On the other hand, due to incompleteness and inconsistency, if we can not find sufficient information proving that the data is correct, it can be considered as of low data quality.

The above observations are drawn from several real world data sets. To make them more concrete and operable for choosing data set, I summarized them into three assumptions on the data that this approach takes. First the most part of data set is correct. In other words those datasets that are intentionally generated to mislead systems, e.g. link farms on the Web, are not considered in our case, because it would be easy to find that the whole dataset is useless. Second most knowledge in a dataset has supporting evidence to some extent, e.g. there are some information for colleagues, co-authors and students around a professor to support the claim that the professor has a significant relation with his/her affiliation. Third, the context around a pair is similar to the contexts around other pairs having the same relationship. For example if a professor advises a student, the patterns of interactions between them, e.g. the project or paper they co-worked in, are also

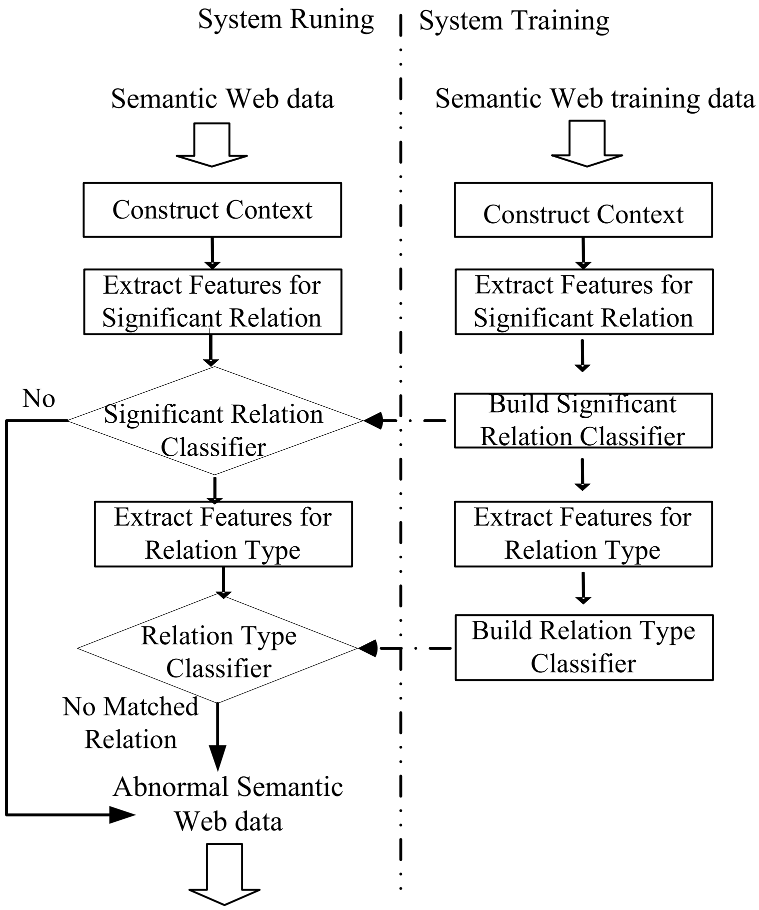

System Runing
System Training
Semantic Web data
Semantic Web training data
Construct Context
Construct Context
Extract Features for Significant Relation
Extract Features for Significant Relation
No
Significant Relation Classifier
Build Significant Relation Classifier
Extract Features for Relation Type
Extract Features for Relation Type
Relation Type Classifier
Build Relation Type Classifier
No Matched Relation
Abnormal Semantic Web data

Figure 4.1: The work flow of the system under closed world assumption.

expected to exist between other pairs of advisor and advisee.

Given the above initial thoughts of the design of this approach, Fig. 4.1 shows the work flow of this approach. The general process is as follows. For each triple, first, the system builds a context. Second, the system uses a classifier to test if there is a significant relation between the two instances in the triple. Third, if previous step shows that there exists a significant relation, then the system matches the patterns in the context with contexts learned from the training data set. The classifier and the pattern matching process are both based on unsupervised learning on the training data set. The process of learning classifier is somehow a reverse process of using them. It takes in every piece of data in training data set and builds the context for it. It extracts features from the context and uses them to build the classifier. All these steps are designed under the effect of closed world assumption. In the first step of constructing context, the approach essentially assumes only the explicit connections between a pair of nodes are contextual support information. In the second step, the closed world assumption is also a foundational element for the definitions of features for classifying pair of instances, because these features are defined to measure the explicit connections demonstrated in the context. It means that it does not take into account the possibility of implicit information that is missing due to incomplete description of the data. Similarly, the last step determines the type of relationship by matching the existing patterns in the context.

4.2 Context Construction

My approach is based on the context around a piece of data. Because the context is used for retrieving features to build a classifier detecting relational errors between the subject and the object, the context should include entities that have certain direct or indirect relationships with the pair of instances in a triple. The detailed definition of a context is in Definition 4. Taking the example graph in section 2.1.2 (re-shown here for convenience), the example context for the triple $< A2, hasTopic, SemanticWeb >$ consists of all the nodes and edges involved in the set of Paths between $A2$ and $SemanticWeb$ whose length is less than the limit. This section discusses how to build the context.

Definition 4. *Given an RDF graph G:=(I,L,R,E), a context of size n for a triple* $< sub, pred, obj >$ *is a function* $C_{G,n} : I \times R \times I \rightarrow \mathcal{G}$. *It produces a subgraph* S *of* G *such that* $S := (I', E', R'), I' \subseteq I, R' \subseteq R, E' \subseteq E, and\ \forall e \subseteq E', \exists p, p \in Paths(sub, obj, G), Length(p) \leq n,$ *where* $e \in Edges(p)$.

On the one hand, the context to be constructed should potentially reinforce the relation between the two instances in a triple to some extent. On the other hand this context also should be similar to contexts around other entity pairs having the same relation. Owing to the small world phenomenon [70], almost every pair of

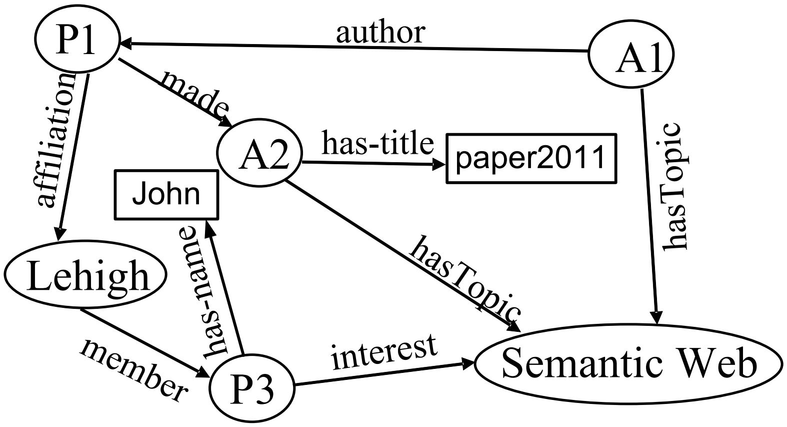


Figure 4.2: Examples of definitions.

entities could be easily connected by a few links. Thus investigating a context just by counting the links is clearly insufficient. Then the question is transformed to what the differences are between contexts around a pair that has a significant relation and a pair that is not related. Two small examples drawn from SWRC dataset are shown in Fig. 4.3. From the graph on the right hand side, we can see that the key connections are through a popular node, *U.S.*. This node is not special for their relation since this node is also connected via the same relation with many other nodes, few of which are involved in this subgraph as well. On the contrary, in the left example, many nodes are distinctive for the pair, such as colleagues and co-authored papers, because a majority of neighbors of these colleagues and papers are involved in the context. Specifically, all the neighbors of the node Li-ding are involved in the actual context for the left hand side, while less than 5% neighbors of it are involved in the actual context for the right hand side. A similar situation exists for the node Deborah-mcguiness. Besides that the nodes are involved differently, the predicates are also used in different ways. Most predicates on the left are used among different instances. While in the right subgraph, each predicate is used on fewer instances, e.g. *U.S.* is the only subject of property *based_near* (property number 9). Therefore we can see that the same instance and predicate have different significance in the contexts for different data, which means it is possible to use them to differentiate the contexts for different data.

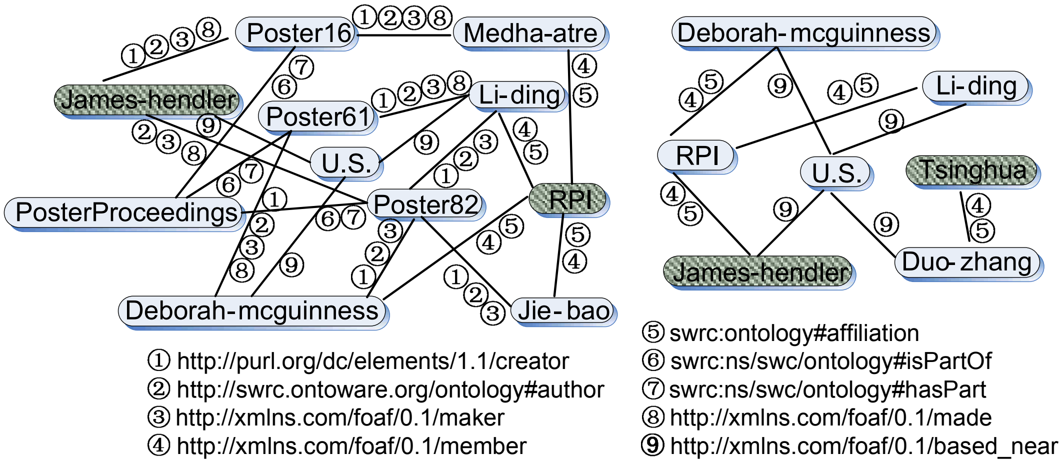

Poster16
Medha-atre
James-hendler
Poster61
Li-ding
U.S.
PosterProceedings
Poster82
RPI
Deborah-mcguinness
Jie-bao
Deborah-mcguinness
Li-ding
RPI
U.S.
Tsinghua
James-hendler
Duo-zhang
① http://purl.org/dc/elements/1.1/creator
② http://swrc.ontoware.org/ontology#author
③ http://xmlns.com/foaf/0.1/maker
④ http://xmlns.com/foaf/0.1/member
⑤ swrc:ontology#affiliation
⑥ swrc:ns/swc/ontology#isPartOf
⑦ swrc:ns/swc/ontology#hasPart
⑧ http://xmlns.com/foaf/0.1/made
⑨ http://xmlns.com/foaf/0.1/based_near

Figure 4.3: Part of two context subgraphs from SWRC dataset, the left hand side is around James Hendler and RPI and the right hand side is around James Hendler and Tsinghua University. The namespace of swrc is <http://data.semanticweb.org/>.

Since the context is a connection subgraph which should contain sufficient information for evaluation, the best way to build such a context is to extract all paths connecting the pair. Thus greedy algorithms [33, 80] that only pick several connections are not suitable here. To avoid an infinite number of paths, cycles are not allowed. Since breadth-first search has exponential space complexity in order to remember all visited nodes, I use depth-first search. The parameter of this algorithm is the max depth limit d, i.e. any path at most consists of d relations among $d + 1$ different consecutive objects along it. Because there is a depth limit, the algorithm does not To find more explicit relations, we treat two paths as different if the predicates are different, including the inverse property, even if the nodes on them are all the same.

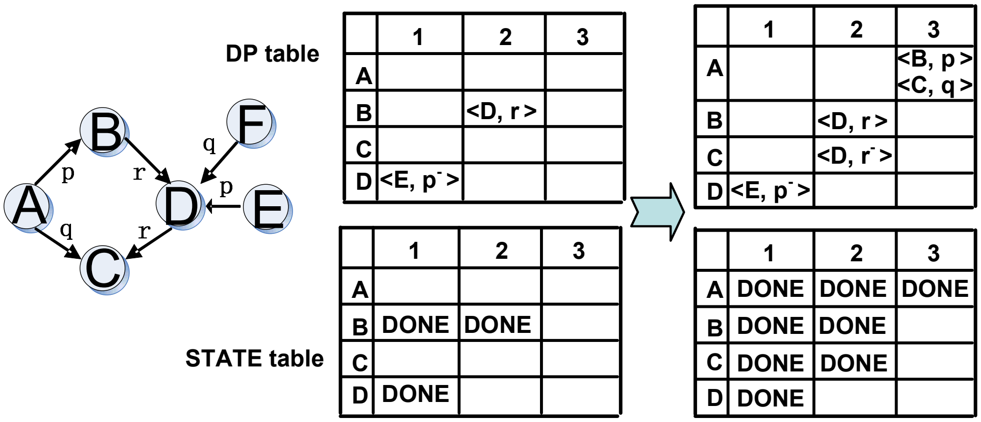
To the best of our knowledge, there is no well-known algorithm for constructing this kind of connection subgraph. Algorithms for finding the shortest paths between all pairs of vertices are not applicable here, because we want to find *all* paths, not just the shortest, between a single pair. Solutions for classic graph reachability are also not appropriate since they only return a binary answer about the reachability between two nodes. Since they do not record the nodes and edges that connect them, it is impossible to know how they are connected and the computation for a pair of nodes can not be reused later for other pair of nodes on the same graph. Bidirectional search is also an option, but it would need to keep the two sets of size $b^{d/2}$ each (b is the branching factor). So in addition to expansion cost, the extra complexity to get their intersection is $b^{d/2}$. Thus when d is small, the time efficiency saved during expansion is traded off by its postprocessing of subgraph building. Therefore to better deal with scalability, we create a bottom-up dynamic programming mechanism which maintains a table recording all computed subpaths leading to the destination. Because only the nodes that can lead to the destination have entries in the table and only the neighbors of these nodes that can lead to destination are recorded, the table is very sparse. Also there is no postprocessing since the table can be viewed as an encoding of the result subgraph.

Algorithm 1 $get\_Context(front, dest, maxd, depth)$, $front$ is the frontier of the current path; $dest$ is the destination; $maxd$ is the max length of a path; $depth$ is the distance from the frontier to the source.

```
$dist2dest \leftarrow maxd - depth$
if $depth < maxd$ then
    for each edge $< front, prop, child >$ or $< child, prop, front >$ do
        if $child = dest$ then
            $DP[front, 1] \leftarrow DP[front, 1] \cup < dest, prop >$
        else
            if $STATE[child, dist2dest - 1] \neq DONE$ then
                $getContext(child, dest, maxd, depth + 1)$
            for each $d, s.t.\ 0 < d < maxd\ and\ DP[child, d] \neq \emptyset$ do
                $DP[front, d + 1] \leftarrow DP[front, d + 1] \cup < child, prop >$
    for each $d, s.t.\ 0 < d \leq dist2dest$ do
        $STATE[front, d] \leftarrow DONE$
return $DP$
```

The Algorithm 1 is the detail of the bottom up dynamic programming algorithm. I used two data structures for supporting this algorithm. They are DP table and STATE table. Both DP and STATE tables are indexed by vertex and distance to the destination. Each cell of DP table has a list of this node's neighbors that can lead to destination with certain distance. Each neighbor in this list is associated with a description of predicate usage on this link. Each cell of STATE table is a boolean value showing whether this neighbor has been explored on a give number of step. The algorithm starts from the source as the frontier with depth zero, though there is no difference which end of the pair is the source. During the search, it treats the same neighbor connected by different predicates as differing neighbors in order to get all possible connections. The algorithm iteratively expands each neighbor of the current node. There are several situations when expanding neighbors. First (line 4-5), if this neighbor is the destination, it records the current node as being one step from the destination. Second (line 7-8), if the STATE table shows that this neighbor with the max distance away from destination is not expanded before (is not equal to DONE), the algorithm expands it. After this neighbor is expanded (line 10), record the subpaths from this neighbor as subpaths with one more step starting from the current frontier. After all neighbors of the current frontier are expanded (line 11-12), it sets the state of it as DONE. The algorithm traverses all nodes once and the dynamic table records every subpath connecting an input pair, so it correctly returns all paths between them. The extracted context subgraph will be our primary input for later steps, because it carries sufficient entities and relationships related to the pair of instances being investigated.

Fig. 4.4 shows two internal states when finding all paths from vertex A to E with maximum path length as three. Both DP and STATE tables are indexed by vertex and distance to the destination. Each cell of DP table has a list of this node's neighbors that can lead to destination with certain distance. Each node in this list is associated with a predicate and a direction showing how it is connected. For example, the STATE table on the left shows that all paths from node D within one step away from the destination are explored. The cell [D, 1] in the DP table means D connects E in one step (i.e. directly) through the predicate of inverse of property p. The cell [B, 2] means that B can finally connect to the destination E in two steps and the immediate step is through node D with predicate r. The DONE in cell [D, 1] in the STATE table means that all the nodes around D in one step has been explored. The DONE in cell [B, 2] in the STATE table means all the nodes within two steps away from node B has been explored. Then when another path through A



	1	2	3
A			
B		<D, r >	
C			
D	<E, p^- >		

	1	2	3
A			
B	DONE	DONE	
C			
D	DONE		

	1	2	3
A			<B, p > <C, q >
B		<D, r >	
C		<D, r^- >	
D	<E, p^- >		

	1	2	3
A	DONE	DONE	DONE
B	DONE	DONE	
C	DONE	DONE	
D	DONE		

Figure 4.4: Example transition of internal states from when B is finished expanding to when A is finished expanding.

and C encounters D, the algorithm looks up the table and knows that D has a path to the destination, then it records the links from C to D and does not expand D again. Similarly the DP table on the right shows that there are two paths from A via B and C respectively to the destination with distance three. Going through these neighbors on the table we can get all paths. Experimental results show that this algorithm is about 30% more efficient than naive depth-first search on the DBpedia dataset. The naive depth-first search that I compared is just my algorithm without using the DP and STATE table. Specifically, the total time of constructing contexts for 1000 triples in SWRC data set with and without using the data structures are 3.1 minutes and 4.3 minutes.

4.3 Link Prediction

The system is designed to measure the certainty level for every piece of information, i.e. a triple or more specifically a certain relationship between two resources in the triple. The measurement of the relationship is based on features extracted from other contextual information. So this problem is similar to the link prediction problem in Web Mining, Social Network Analysis (SNA) to a certain extent. So I reviewed the link prediction problem first.

The link prediction problem in SNA can be formalized as follows. Given a snapshot of a social network, how to infer which new interactions among its members are likely to occur in the near future? The link prediction problem is also related to the problem of inferring missing links from an observed network: in a number of domains, one constructs a network of interactions based on observable data and then tries to infer additional links that, while not directly visible, are likely to exist. From the perspective of our problem, these two lines of work are both similar and helpful for solving our problem. Liben-Nowell et al. [60] and Getoor et al. [37] both reviewed many link prediction techniques, which are discussed for networks of a single type of links, e.g. co-authorship network. To the best of my knowledge, there is no existing well-known works focusing on networks of large amount of types of links. Thus I mainly review the methods they discussed here. Most of the methods they compared assign a connection weight $score(x, y)$ to pairs of nodes $< x, y >$, based on the input graph G_{collab}, and then produce a ranked list in decreasing order of $score(x, y)$. Thus, they can be viewed as computing a measure of proximity or "similarity" between nodes x and y, relative to the network topology. Perhaps the most basic approach is to rank pairs $< x, y >$ by the length of their shortest path in G_{collab}. Such a measure follows the notion that collaboration networks are "small worlds," in which individuals are related through short chains. Other techniques can be categorized in the following two groups.

The first group of works is the methods based on node neighborhoods.

Common neighbors. The most direct implementation of this idea for link prediction is to define $score(x, y) := |\Gamma(x) \cap \Gamma(y)|$, where the set $\Gamma(x)$ consists of the neighbors of the node x in G_{collab}. So this metric is to measure the number of neighbors that x and y have in common. Newman [74] has computed this quantity in the context of collaboration networks, verifying a correlation between the number of common neighbors of x and y at time t, and the probability that they will collaborate in the future.

Jaccard's coefficient and Adamic/Adar. The Jaccard coefficient - a commonly used similarity metric in information retrieval [84] measures the probability that both x and y have a feature f, for a randomly selected feature f that either x or y

has. If we take "features" here to be neighbors in G_{collab}, this leads to the measure $score(x, y) := |\Gamma(x) \cap \Gamma(y)|/|\Gamma(x) \cup \Gamma(y)|$. Adamic and Adar [2] consider a related measure, in the context of deciding when two personal home pages are strongly "related." To do this, they compute features of the pages, and define the similarity between two pages to be $\sum_{z: features\ shared\ by\ x,y} \frac{1}{log(frequency(z))}$.

Preferential attachment. Preferential attachment has received considerable attention as a model of the growth of networks [71]. The basic premise is that the probability that a new edge involves node x is proportional to $\Gamma(x)$, the current number of neighbors of x. Newman [74] and Barabasi et al. [12] have further proposed, on the basis of empirical evidence, that the probability of co-authorship of x and y is correlated with the product of the number of collaborators of x and y. This corresponds to the measure $score(x, y) := |\Gamma(x)| \cdot |\Gamma(y)|$.

The other group of works is the methods based on the ensemble of all paths.

Katz. Katz defines a measure that directly sums over this collection of paths, exponentially damped by length to count short paths more heavily. This leads to the measure $score(x, y) := \sum_{l=1}^{\infty} \beta^l \cdot |paths_{x,y}^{<l>}|$ where $paths_{x,y}^{<l>}$ is the set of all length-l paths from x to y. (A very small β yields predictions much like common neighbors, since paths of length three or more contribute very little to the summation.) One can verify that the matrix of scores is given by $(I - \beta M)^{-1} - I$, where M is the adjacency matrix of the graph and I is the identity matrix. There are two variants of this Katz measure: (1) unweighted, in which $paths_{x,y}^{<l>} = 1$ if x and y have collaborated and 0 otherwise, and (2) weighted, in which $paths_{x,y}^{<l>}$ is the number of times that x and y have collaborated.

Hitting time, PageRank, and variants. A random walk on G_{collab} starts at a node x, and iteratively moves to a neighbor of x chosen uniformly at random. The hitting time $H_{x,y}$ from x to y is the expected number of steps required for a random walk starting at x to reach y. Since the hitting time is not in general symmetric, it is also natural to consider the commute time $C_{x,y} := H_{x,y} + H_{y,x}$. Both of these measures serve as natural proximity measures, and hence (negated) can be used as $score(x, y)$.

SimRank. SimRank is a fixed point of the following recursive definition: two nodes are similar to the extent that they are joined to similar neighbors. Numerically, this is specified by defining $similarity(x, x) := 1$ and $similarity(x, y) := \gamma \cdot \frac{\sum_{a \in \Gamma x} \sum_{b \in \Gamma y} similarity(a,b)}{|\Gamma(x)| \cdot |\Gamma(y)|}$

Some SNA researchers begin to explore the help of Semantic Web technologies, such as using ontologies, e.g. Caragea et al. [22], but they use the ontology as a dictionary to help determine the distance between concepts mentioned in users' interests and still only predict the single friendship. The link discovery in multi-relational data by Lin et al. [61] tried to find novel interesting paths between entities, which is rarely, interestingly linked entities in multi-relational data sets, rather than a normal link prediction. Tag prediction is a new research topic. From the graph point of view, if we take the tags placed by people on documents as the labels of the links between the people and the documents, this graph is more similar to our RDF graph than other social network. The different tags on the same document labeled by different people certainly have similarities, because they are all based on the content of that document. Therefore, a lot of language model based methods are explored to predict the tags.

4.4 Credible Relation

Most approaches in the domain of link predication on SNA cannot be directly adapted to the problem of evaluating relationships between nodes on RDF graphs, although many thoughts there are very helpful and inspiring. The entity types in the domains of link prediction on SNA are more limited, such as web pages or persons and the link types too, such as hyper links or friendships. However the Semantic Web consists of much more heterogeneous instances and semantic links. Therefore it needs more elegant and complex solutions to take into account special characteristics that only the Semantic Web has. As introduced in section 4.3, the most popular methods in the domain of SNA are the following [60]. First, rank pairs $< x, y >$ by the length of their shortest path. Second, rank pairs $< x, y >$ by the number of neighbors that x and y have in common. Third, Jaccard's coefficient, a commonly used similarity metric in information retrieval, measures the probability that both x and y have a feature f, for a randomly selected feature f that either x or y has. Fourth, Adamic/Adar computes features of the pages and defines the similarity between two pages to be $\sum_{z: features\ shared\ by\ x,y} \frac{1}{\log(frequency(z))}$.

It can be observed that these popular methods used in Web Mining and SNA mainly consider two aspects: the characteristics of link graph structure and characteristic of the node itself. The reason for the first aspect is that there are just few (if not only one) link types on those networks. Compared to it, the Semantic Web consists of much more types of relationships connecting entities on the network and so the combination of these relationships can give more variations of semantics. For the second aspect mainly considered by link prediction in SNA, the instances on these networks are relatively more independent, a web page may have no hyperlinks both pointing to and pointed by other pages and a person can have no friends on a friendship network. Because they have much information that is associated with itself, e.g. the profile or browsing history of a user or the text content on the webpage, many useful features can still be retrieved within the node itself. However the instances on the Semantic Web are denoted by URIs which do not contain much useful information on their own and all the descriptions of them are conveyed by the links with other instances or values in the network. To be complete and meaningful, the knowledge of every node is more dependent on connections with other instances through edges in the network. In other words, nodes are meaningless if no links connect them on the Semantic Web. To conclude from these two comparisons, the viable features on the Semantic Web might need to combine the graph structure and the link types over them.

Following the above conclusion through comparing and contrasting link analysis tasks on two networks, I propose to take into account the following three features on the link analysis on the Semantic Web. First, the link structures in the subgraph that is around the pair of nodes to be evaluated is still important. But different from the SNA domain which often only investigates very small neighborhoods, the link structures should contain enough attributes describing these neighboring instances and therefore need several more hops than just one or two. Second, in the subgraph that is around the pair to be evaluated, the characteristics and differences among predicates should be explored, because predicates are one of the most important semantic features. Third, axiomatic inference is the essential capability of the Semantic Web knowledge. But much real world knowledge can not be represented using logic axiom of crisp, monotonic predicates, e.g. $smoke(x) \rightarrow cancer(x)$. A lot of rules that are even more vague than the example axiom exist potentially in the real world data, e.g. a graduate student often works in the project that his/her advisor is the project investigator. Exploring these underlying rules can greatly improve the probabilistic inference on the data that is needed in our problem. Fourth, the concepts used in describing the information in the context should be explored. Specifically, if the concepts are more specific, the information would be more accurate. The concepts include both class types and property types.

I designed several metrics to measure the differences among extracted subgraphs. Having the context for a triple, the following demonstrates the indicators of a significant relation (I call this component SR) between the pair of instances in a triple. Because numeric features would be the most simple to use in a classifier and the rules using them would also be easy to present, I tried to transform them into numeric spaces.

Class Distinctiveness The indicator Class Distinctiveness (CD) is used to measure the information content of each node. In information theory, the amount of information contained in an event is measured by the negative logarithm of the probability of occurrence of the event. The amount of information gained or uncertainty removed by knowing that a probabilistic variable χ has the outcome x_i is given by $I(\chi = x_i) = -logPr(x_i)$. For any class $c \in C$ where C is the set of all classes, the probability that $\chi = c$ is given by $Pr(\chi = c) = |c|/|I|$, where I is the set of all instances. Then we define the CD as the average of the information content of all the URIs in the subgraph (shown in Definition 5), where $c(i)$ is the class type of instance i. If an instance o has multiple types, e.g. c_1 and c_2, then $|c(o)| = |c_1 \cup c_2|$. Using RDF terms, CD is a measure of the specificity of classes that instances in this subgraph are type of. The intuition is the contexts with more specific concepts are more precise.

Definition 5. *Given a subgraph $S = (I', L', E', R')$ of graph $G = (I, L, E, R)$, the CD of S is defined as*

$$CD(S) = -\frac{1}{|I'|}\sum_{i \in I'} logPr(\chi = c(i)) = -\frac{1}{|I'|}\sum_{i \in I'} log\frac{|c(i)|}{|I|} \tag{4.1}$$

Node Distinctiveness The indicator Node Distinctiveness (ND) is used to measure how the nodes in the subgraph are special to this subgraph. A node is special to a subgraph if it is connected more strongly to nodes in this subgraph than to those outside of the subgraph. Each node's weight is weighted by its usages in subgraph. A node could be the subject or the object for a connection and be special when it is special on either one. So for each instance, we separately compute and average them to reflect the distinctiveness (shown in Definition 6, 7 and 8).

Definition 6. *Given a graph $G = (I, L, E, R)$ and an instance $U \in I$,* ***in-degree*** *of U w.r.t G is defined as $In_G(U) = |\{e|e = (s, p, U)$ and $e \in E\}|$, and similarly* ***out-degree*** *of U w.r.t G is defined as $Out_G(U) = |\{e|e = (U, p, o)$ and $e \in E\}|$.*

Definition 7. *Given a subgraph $S = (I', L', E', R')$ of graph $G = (I, L, E, R)$ and an instance $U \in I'$, the Node Weight (NW) of U w.r.t S is defined as*

$$NW(U, S) = \frac{1}{2}(\frac{In_S(U)}{In_G(U)} \times \frac{In_S(U)}{|E'|} + \frac{Out_S(U)}{Out_G(U)} \times \frac{Out_S(U)}{|E'|}) \tag{4.2}$$

Definition 8. *Given a subgraph $S = (I', L', E', R')$ of graph $G = (I, L, E, R)$, the ND of S is defined as*

$$ND(S) = \sum_{i \in I'} NW(i, S) \tag{4.3}$$

Predicate Distinctiveness The indicator Predicate Distinctiveness (PD) is to measure how special the predicates are with respect to this subgraph (shown in 9, 10 and 11).

Definition 9. *Given a graph $G = (I, L, E, R)$ and a predicate $P \in R$, the number of edges of P is $Edges_G(P) = |\{e|e = (s, P, o)$ and $e \in E\}|$; the number of distinct subjects of P is $Sub_G(P) = |\{s|e = (s, P, o)$ and $e \in E\}|$; the number of distinct objects of P is $Obj_G(P) = |\{o|e = (s, P, o)$ and $e \in E\}|$.*

Definition 10. *Given a subgraph $S = (I', L', E', R')$ and a predicate $P \in R'$, the Predicate Weight (PW) of P w.r.t S is defined as*

$$PW(P, S) = \frac{Edges_S(P)}{Edges_G(P)} \times \frac{Subs_S(P) + Objs_S(P)}{Sub_G(P) + Obj_G(P)} \tag{4.4}$$

Definition 11. *Given a subgraph $S = (I', L', E', R')$, $r_i \in R', 0 < i$, the PD of S is defined as*

$$PD(S) = |R'| \sum_{r_i \in R'} \frac{(PW(r_i, S) \times Edges_S(r_i))}{\sum_i Edges_S(r_i)} \tag{4.5}$$

There are several considerations for designing the PD. First, for each predicate, what percentage of its usages is within the subgraph? Second, how many distinct subjects and objects of each predicate are used in the subgraph? Third, the variety of subjects and objects should be considered together. Because some predicates have very few distinct subjects or objects but many more of the other (e.g. citizenship, the variety of its subject values would make it distinctive if not considering that its object values may be only several). Fourth, the sum of all the predicates are weighted based on each predicate's contribution to number of the edges in the subgraph. Fifth, using the number of distinct predicates as a factor (the leading coefficient $|R'|$ in the formula of PD) can compensate some extreme cases where small context graph with few types of relations makes the CD and ND be very high.

As described before, all the above three measures will be used to differentiate data into different qualities. It is achieved by using a classifier. The problem that a classifier solves can be stated as follows: given training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ produce a rule (or "classifier") h, such that $h(x)$ can be evaluated for any possible value of x (not just those included in the training data) and such that the group attributed to any new observation, specifically $\hat{y} = h(x)$, is as close as possible to the true group label y. Therefore I will use these measures to build a function h. There are multiple detailed solution of h and they depend on the mechanisms of building the classifier. I will compare and test several of them in subsequent section for experiments.

4.5 Patterns of Relation

The component SR determines if there is a significant relation between a pair of nodes on the RDF graph, if the answer is yes, then a second component checks if the type of relationship that is entailed through probabilistic classification using the context agrees with the predicate actually used in the triple. Since the component is to determine the relation type, it is called RT.

Because the entailment is on a per predicate basis and the number of predicates in a data set is much smaller than that of instances or triples, using predicates in the context as features to predict the relation would be an efficient way. In addition, predicate co-occurrence has been frequently used in ontology alignment and coreference resolution. When aligning ontology classes, two classes that have more common characteristics are more likely be equivalent. The common characteristics include the possible properties that a class can have. When resolving coreferenced instances, two instances that have more common property values are more likely be the same. My algorithm also essentially utilizes this similar idea used in ontology alignment and coreference resolution. The idea is that if a context shows patterns of predicate usage that also appeared in contexts around other pairs, then the relations between those pairs are probably similar to this relation.

The input of the algorithm is the context introduced in section 4.2. The patterns for a predicate consist of predicates extracted from contexts for triples with this predicate, but they should not be treated simply as a bag of predicates. The first reason is that the same set of predicates would reflect different meaning if the order of their usages is different. Second, different join conditions among triples convey different interconnecting semantic patterns and relations between two end points. For instance, we have four triples: $< A1, studentOf, B1 >$, $< B1, advisorOf, C1 >$, $< A2, studentOf, B2 >$ and $< C2, advisorOf, B2 >$. The sequence of the first two predicates is the same as that of the last two predicates. But $A1$ and $C1$ are connected because they are both students of $B1$ while $A2$ and $C2$ are connected because $A2$ is academic descendant of $C2$. The two relations are totally different. Considering the points above, we define predicate patterns below. I use traditional

DL inverse property representation to indicate that the triple of this predicate is joined via object with previous triple and via subject with next triple.

Definition 12. *Given a graph* $G = (I, L, E, R)$*, a Semantic Connection is* $c = \langle r_1, r_2, ..., r_n \rangle$*, where* $r_i \in (R \cap R^-)$*, and* $\exists I_0, I_1, ..., I_n$*, such that* $\langle I_0, r_1, I_1, r_2, I_2, \ldots, r_n, I_n \rangle$ *is a Path between* I_0 *and* I_n*.* $Head(c) = r_1$*.* $Tail(c) = r_n$*.The function* $Inst(c, G)$ *returns the set of all such Paths in graph G. Length*(c) *is defined as the number of relations in the tuple.*

Definition 13. *Given a graph* $G = (I, L, E, R)$*, the Inverted Predicate Frequency for a given semantic connection c is defined as*

$$IPF(c, R) = \frac{|R|}{|\{r \in R | \exists < s, r, o > \in E, s.t.\ Inst(c) \cap Paths(s, o, G) \neq \emptyset\}|}$$

Definition 14. *Given a graph* $G = (I, L, E, R)$*, the Semantic Connection Frequency for a given semantic connection c in contexts for triples of given predicate is defined as*

$$SCP(c, r) = \sum_{<s,r,o>\in E} |Inst(c, G) \cap Paths(s, o, G)|$$

Definition 15. *Given a graph* $G = (I, L, E, R)$*, the pattern template for a given predicate pr is defined as* $Patt(pr) = \{(c_i, w_i) | c_i$ *is a Semantic Connection,* $w_i = SCP(c_i, pr) \times IPF(c_i, R)\}$*.*

Shown in Definition 15, the weight for a Semantic Connection is based upon the number of instantiations of the Semantic Connection in the contexts for all triples. If we make an analogy between a predicate and a document class, the Semantic Connections and the terms respectively, the weight for a Semantic Connection is similar to a tf/idf term weight used in information retrieval. In tf/idf, a term is weighted as the number of its usages in a document divided by the inverted frequency of documents that contain the term. So the tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to control for the fact that some words are generally more common than others. Similarly, a Semantic Connection is weighted in proportion of the number of its instantiations in the contexts for triples of the predicate. Meanwhile it is offset by the inverted frequency of predicates that have this pattern. A predicate has a semantic connection, if and only if the contexts for some triples of this predicate contains the instantiations of the semantic connection.

$$pred(t) = \max_{pr \in R} \sum_{<p_i, w_i> \in Patt(pr)} w_i \times match(t, p_i) \tag{4.6}$$

$$match(< s, r, o >, p_i) = \begin{cases} 0, & Inst(p_i, G) \cap Paths(s, o, G) = \emptyset \\ 1, & Inst(p_i, G) \cap Paths(s, o, G) \neq \emptyset \end{cases} \tag{4.7}$$

The previous definitions introduced how the system extract patterns and their weights during learning process. At runtime, the system extracts the Semantic Connections in the context of a triple to be evaluated. Then it matches the extracted semantic connections with the learned semantic connections. Equation 4.6 describes the criterion for the best matched predicate for a triple t. The best predicate has the largest sum of matched connection weights. In equation 4.7, $match()$ is a boolean predicate: whether the triple's context has this pattern and w_i. The matching complexity is Θ(mn), where m is the number of connections from the context of the triple to be evaluated and n is the total number of connections in referenced data set.

4.6 Experimental Setup

In our experiments, we selected the SWRC data set which has 100K triples and 67K resources and the DBpedia infobox data set which has 10M triples and 3M resources. They are widely used and from different domains. The SWRC data set generically models key entities relevant for typical research communities and the relations between them. DBpedia is a data set containing structured content extracted from the information created as part of the Wikipedia project.

Because the referenced data is assumed generally correct, when training, the system uses some existing triples for each predicate from the referenced data set as positive examples and any triples neither in nor entailed by the original data set can be used as negative examples. When testing, for each predicate, I pick 200 random triples which are not in training set as positive test examples. Because almost no data sets have the negation of triples (recently OWL 2 [1] added this function), the negative triples used in test are generated through the following process. For each predicate used in positive examples, we create a domain set consisting of all the distinct subjects of positive example triples using this predicate and similarly a range set consisting of all objects from them. Then a subject and an object from each set are randomly selected to compose a synthetic triple of this predicate. This step can ensure that the synthetic triple still conforms to the ontologies of this data set. Otherwise it would be trivial to find that it is suspect. Finally if the generated triple is not entailed by the original data set, it qualifies as a negative example. I believe SWRC and DBpedia to be highly reliable and complete data. To verify the reliability of test set, four Semantic Web experts verified 100 randomly sampled triples from SWRC data set and 100 randomly generated negative examples. They verify the data by using a simple interface through which they can explore relevant triples in the knowledge base and Sindice[2], a popular Semantic Web search engine. The experts verified that all the positives are correct and all the negatives are incorrect.

The training process is to establish the parameters of the classifiers in the two components. The process to get the pattern weights used to entail the predicate is introduced in section 4.5. To get the weights of indicators for the existence of a significant relation, I compute the three indicator values (CD, ND, PD) for every training triple and put them into a classifier as feature values of these triples. To avoid bias, I removed the original direct links between the pair of objects in positive triples for all experiments so that both positive and negative triples are unknown to the system.

The experiments are primarily designed to check if the system can make distinctions between ordinary triples and abnormal triples in the test set. Component SR

[1] http://www.w3.org/TR/owl2-overview/

[2] http://www.sindice.com

	decision tree	naive Bayesian	KNN (k=5)	KNN (k=10)	BLR
Precision	90.2%	88.6%	90.5%	90.2%	89.4%
Recall	72.8%	72.9%	71.6%	71.7%	72.9%
F-measure	80.6%	80.0%	79.9%	79.9%	80.3%

Table 4.1: Comparison among different types of classifiers.

and RT are separated to test if each functions well.

4.7 Results

First of all, I did the experiment to determine which kind of classifier is best suitable to the SR component. I selected 1000 random triples from SWRC data set and 1000 random generated negative triples for training. Similarly I selected 200 random triples from SWRC data set and 200 random generated negative triples for testing. I compared the performance of several popular classifiers, such as decision tree, naive Bayesian, kth nearest neighbor and binomial logistic regression (BLR). Overall, the decision tree has the best F-measure, though others are not far behind. I believe that the reasons why the recall is relatively lower than precision are as follows. These features captured the common characteristics of positive examples, so true negatives are hard to show these characteristics. Therefore, the precision is high. However, due to the nature of real world data, some true positives might not strongly show these characteristics, thus they are easier to be classified as negatives. Therefore, the recall is relatively lower than the precision. I believe it is important and good that the system does not classify many negatives as positives. As a result, the system does not lose the opportunity to detect true negatives in later steps. In all the following experiments, I use decision tree as the classifier for the SR component.

Before the experiments on system accuracy, I checked the running time of system. Because the dominant part of execution time is for the context construction. Table 4.5 shows the total construction time for 1000 triples in SWRC when context size varies.

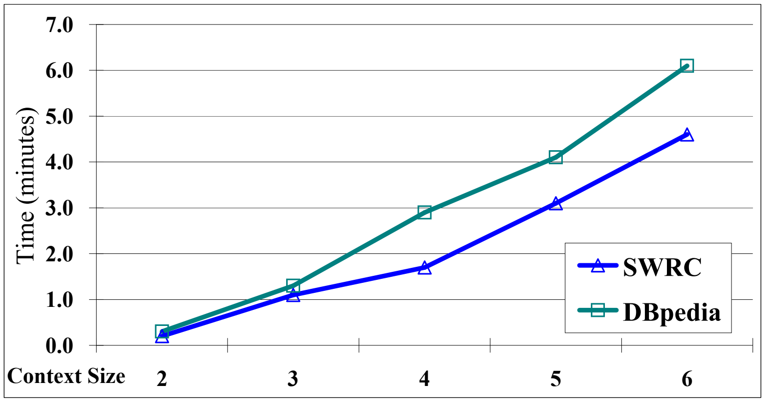


Figure 4.5: The effect of context size on context construction time.

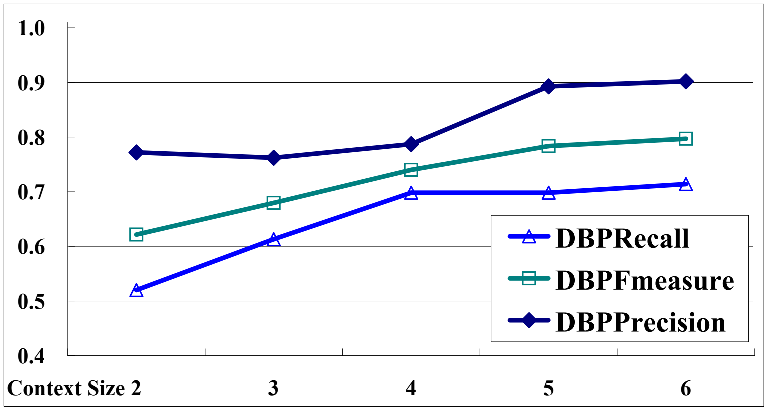


Figure 4.6: (a)(b) Results of the component SR on determining significant relations on SWRC.

The first group of experiments show the effect of context size (defined in Definition 4) on determining the existence of significant relations. Both test sets consist of an equal number of positive and negative examples. The results reflect a similar trend on both data sets (Fig. 4.6 and Fig. 4.7). The precision does not drop much when the context size decreases. The reason is that when the subgraph is smaller (resulting in less context information), the links in the contexts of both positive and negative samples are easily broken in negative contexts. So it would be harder for the negatives to have a well clustered supporting evidence that is necessary to be classified as positive. Thus the false positives become fewer when the context shrinks. Similarly due to the removed links in the contexts for positive examples,

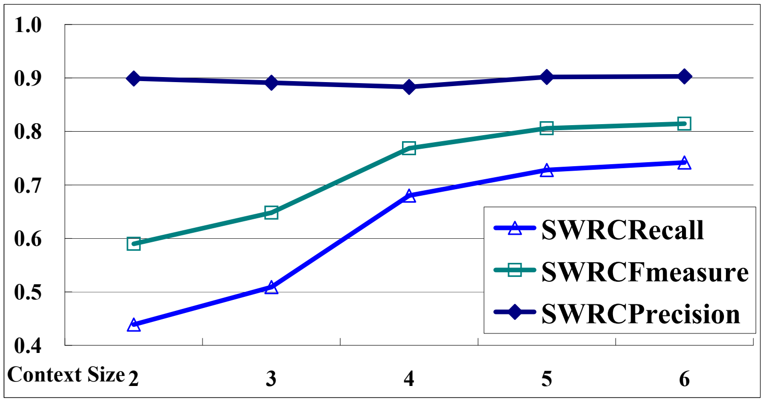


Figure 4.7: (a)(b) Results of the component SR on determining significant relations on DBpedia.

some of the positive examples will lose some clues that are useful for the system to classify them as positive when the context shrinks. So the number of false negatives increases and the recall drops more than precision. In addition, we notice that the improvement on recall on the SWRC data set is more than that on the DBpedia data set when context size increases. I believe the reason is that SWRC data set has more relational descriptions among instances, specifically the average density (number of edges divided by square of number of nodes) of context graphs on SWRC data set is around five times of that on DBpedia data set. So the contexts in SWRC gains more descriptions when the path length is larger. Another observation that can be an important reason is that SWRC data set have more redundant descriptions, i.e. property usages, for many instances on the domain. It can be noted that when the context size is greater than five, both the precision and the recall gains little. Then I compared the number of nodes and edges in the contexts of size five and six and observed that they both do not increase much. Specifically, the increase on the number of nodes from five to six is less than 10% and the number of edges from five to six is less than 8% on two data sets. However the increase of computation time is more than 24%. So the performance didn't gain much due to similar contexts while efficiency is lost much when using big context size. Therefore in later experiments, I set the context size five in default.

To check how the system relies on the two assumptions on the reference data,

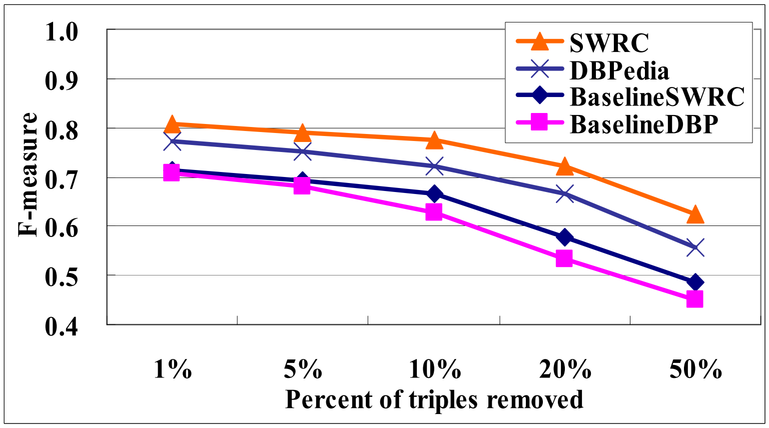


Figure 4.8: Impact of less complete data on the systems ability to detect significant relations.

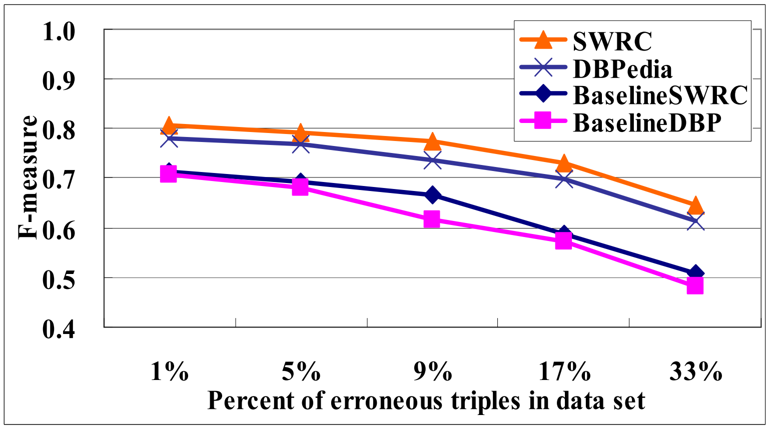


Figure 4.9: Impact of erroneous data on the systems ability to detect significant relations.

i.e. the closed world assumption and the referenced data set is generally correct, the second group of experiments checks the system performance when some random triples in the reference data are removed (Fig. 4.8) and when the reference data has some erroneous triples (Fig. 4.9). In Fig. 4.8 I removed 1%, 5%, 10%, 20% and 50% of data, respectively. In Fig. 4.9 I added erroneous triples with the same amount as the removed triples in previous experiment, which generates a data set with 1%, 5%, 9%, 17% and 33% erroneous data, respectively. Removing triples gives two aspects of impact to the system. One is that less context information will be provided for all triples and the other is that some missing triples that the system assumes incorrect are factual. We see that when 10% triples are removed, the system still can give decent performance (drops within 5% and it occurs on DBpedia). Similarly when 10% of triples are incorrect, the performance only drops within 4% (it occurs on DBpedia). Comparing Fig. 4.8 and 4.9, the effect of erroneous triples is not as much as that of the triples removed. The reason is that the learning is based on the agreement among the majority of the data. Some erroneous triples would probably incur some patterns that others hardly agree with, while removing triples makes many agreed patterns disappear or become blurred. Since our component SR for determining significant relation is similar to the link prediction in SNA, we also compared with the following baseline classifier. Among popular link predictors in SNA, such as Jaccard, Katz weighted, Katz unweighted, common neighbors and preferential attachment [60], we observed that the baseline using Katz weighted and preferential attachment is almost as good as combining all above popular link predictors and so the baseline in Fig. 4.8 and 4.9 uses these two predictors in the same classifier. From the figures, we can see that my system using the proposed measures is much better than the baseline using popular link predictors.

The third group of experiments is to check the component SR with different subsets of indicators (shown in Fig. 4.10). We see that only using the ND can get better performance than the other two on SWRC, while only using the PD or CD is better in DBpedia. Combinations of two indicators are better than using single ones and but no combination is dominant across all domains. Finally the combination of the three is the best. It proves that three indicators capture different significant aspects of a context.

In the last group of experiments (Fig. 4.11), we tested the component RT to check if the Semantic Connections are useful to determine the relation type between the pair of objects in a triple. If the system can accurately determine the best predicate for their relation, the system can also differentiate the triple using the correct predicate from that using an incorrect predicate. In this experiment, the positive results from the SR are then used to evaluate the correctness of the predicate. Thus we should note that the performance of this component is affected

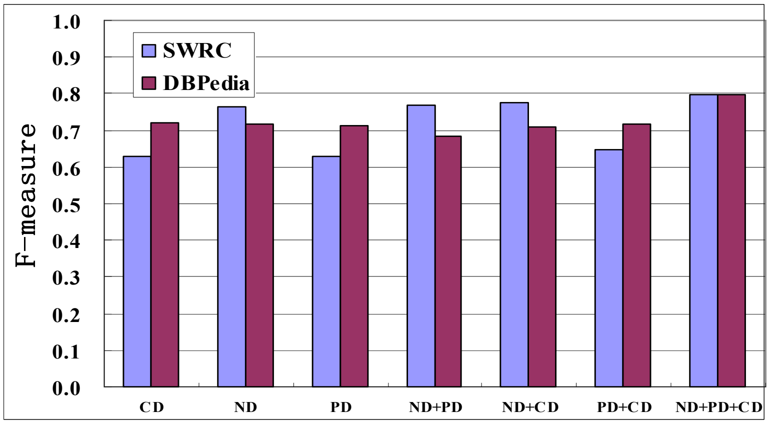


Figure 4.10: Results of the component SR using subsets of indicators on two data sets.

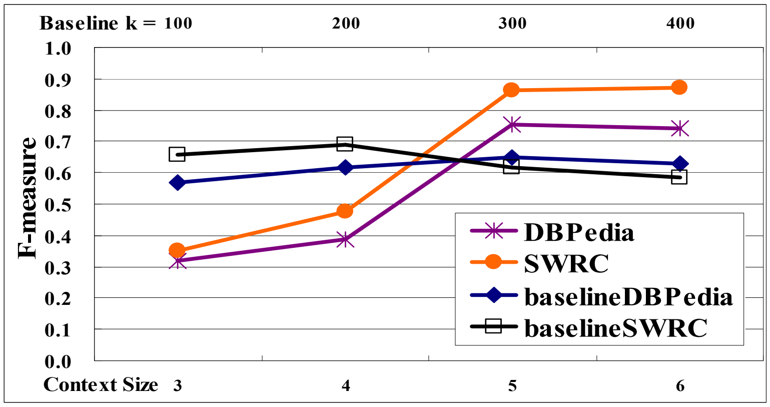


Figure 4.11: Comparison between component RT and the baseline on determining relation types on two data sets.

by that of SR. In order to understand our system's capabilities, we compare it to a baseline system based on predicate suggestion systems. For instance, such systems usually find similar instances and suggest some predicates used on those instances but not on this instance [75]. If both instances in a triple have a group of similar instances respectively, the predicates connecting the two groups of instances could also be the predicates between them. So the baseline system is built by finding a set consisting of the top kth similar instances for each object and then picking the top ranked predicate connecting these two sets of instances as the predicted relation.

The similarity between instances is measured by counting the number of the same predicate and object pairs. In this experiment, I am not implying the values of k= 100, 200 and 300 are comparable to context sizes of 3, 4 and 5. I am showing the trend of each system and comparing the best performance of each of them when variable of each changes. The figure shows that although RT performs worse than the baseline for context sizes of three and four, when the context size is bigger than five, it performs better than the best configuration of the baseline.

In summary, the major observations from these experiments are as follows. The two-step system indeed captures some characteristics underlying the context of every piece of data that can help to differentiate their normality. With bigger context sizes, the overall accuracy improves. But when the context size is too large (bigger than five), there are diminishing returns in accuracy while the execution time increases rapidly. When using the referenced data as training, the density of the data has more impact than general correctness, because comparing two situations, the performance drops more when I removed a significant portion of data from it. This removal operation essentially breaks the approximate closed world assumption which is the essential foundation of the algorithm used in this system.

Chapter 5

Data Correctness under the Open World Assumption

In my previous work for the scenario where the closed world assumption is applied to Semantic Web data, I learned that there are indeed some potential common characteristics underlying the data that can help to detect quality issues. However, the closed world assumption does not usually apply to the Semantic Web. Further through results, I observed that there is a significant performance drop when this assumption does not hold, i.e. when I removed a significant portion of data from the training data set, thus an approximate closed world assumption is invalid. Then in this chapter, I address the research question whether there is a way to discover similar useful patterns in a Semantic Web data set where the open world assumption applies. The open world assumption is the assumption that the truth-value of a statement is independent of whether or not it is known to be true. The essential idea in the previous chapter can inspire the approach in this chapter. I still try to utilize patterns to measure (dis)similarity between the data in order to detect abnormal Semantic Web data [104]. I no longer require either the data in question or the training data to be close to the state of the closed world assumption, i.e. for some objects in the domain, the values for some properties that are applicable to these objects might not be given.

In my previous work, I designed a two-step system for detecting abnormal Semantic Web data. There are drawbacks that can be improved. First, although the decomposition of system into two components: link existence checking and link type checking, is natural, if we deem no relation as a special type of relation, we can integrate them into one by only checking the type of relation. That will lead to both more efficient computation and a more concise system architecture. However it requires to improve the functions of the relationship type classifier in my previous work and so it is necessary to improve the patterns both in extraction and computation used for determining the relation type. Focusing on those patterns, the second drawback of previous algorithm is as follows. When I was investigating the semantic connections between two instances, I did not take into account the differences among predicates, in short the semantics. In other words, if there are the same number of instantiations for two semantic connections, these two semantic connections should also differently affect the determination of a relation type if any predicates on two semantic connections are different, because their semantics are different. To this end, it essentially requires to compute the discriminative weights of predicates in order to affect the weights of semantic connections.

There will be two important changes when the closed world assumption is not valid. First, I can not assume the triples not present in the training data set as false, thus I cannot use them as negative samples for training. Therefore I need to adjust the learning model to deal with this situation. Second, when some of the data to be evaluated is not comprehensively described, i.e. there could be very few semantic connections between the two objects in a triple, I need an approach to find more supporting evidence for the pair: essentially an expansion of its context. Corresponding to these two changes, I designed a context representation model and learning model that takes into account requirements corresponding to each changes. The work flow of this system is shown in Fig. 5.1. The context constructed in the step before the expansion essentially is the same as the context constructed in my previous system. However the representation is no longer in a subgraph form but a vector space model consisting of semantic connections. This new representation will make the computation of pattern weights and later pattern matching more

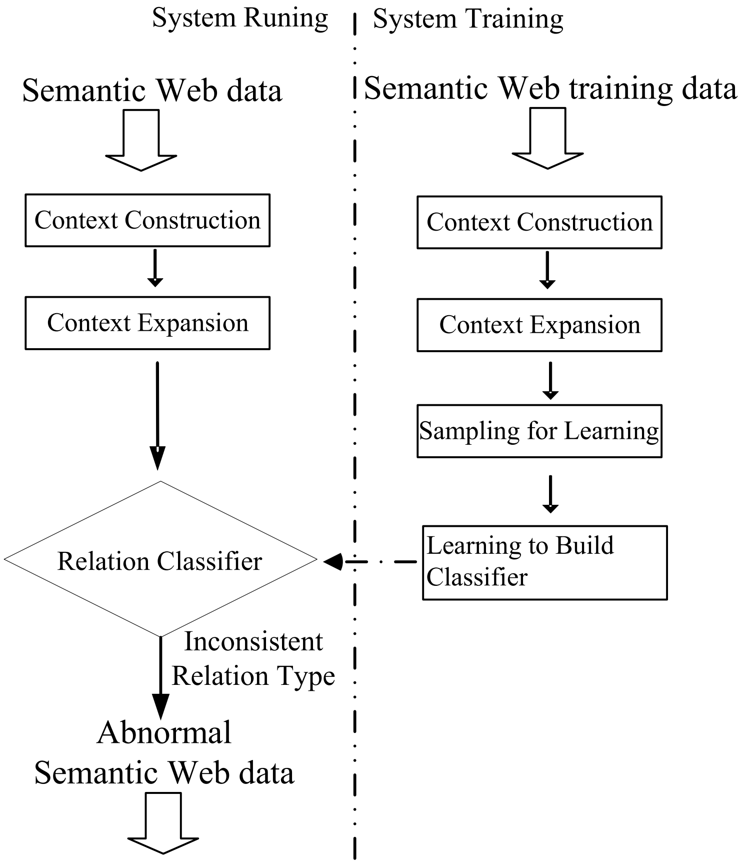


Figure 5.1: The work flow of the system under open world assumption.

convenient. Using the representation model and after the expansion, the contexts are used in the unsupervised learning through a learning model that is adapted from tag prediction problem to this problem.

Like the algorithm in Chapter 4, this work also utilizes the idea of classification to differentiate data into normal or abnormal state. From the classification perspective, they have the following differences. The classifier used in the SR component of my previous work in Chapter 4 is a classifier that determines existence of credible relation between two instances. I used a decision tree as the classifying algorithm and use three numeric distinctiveness metrics as features. The classifier used in the RT component of my previous work in Chapter 4 is built by input both the features, i.e. patterns, and the algorithm to classify using these patterns, i.e. the scoring function of pattern matching. The classifier in this chapter is adapted from its original usage in tag prediction domain. It is more customized for this problem than naive Bayesian, decision tree and some other general ones. Further both the patterns and the classifying algorithm are different from the one used in the my previous work in Chapter 4. Specifically, the patterns in this work will be extracted from extended context of a triple as opposed to immediate context and their representations are significantly different from those in my previous work. Finally, given the new patterns, the classifying algorithm in this work will consider the open world assumption. The overall system architecture is shown as Fig. 5.1.

5.1 Context Representation Model

Considering the above analysis of problem requirement and thoughts on algorithm design, I begin with the core part of the classifier to be built, i.e. the scoring function, and follow with details of how to compute it and how to get the required information. Formally, my problem is defined as: given the pair u of subject s and object o, how significant is some relation p between the pair u (written $y_{u,p}$). This is the scoring function that determines which relations appear to be normal for the pair of instances and which are not. Let U_p be the set of all pairs that have the relation p and u' be any pair of subject s' and object o' ($s'! = s$ or $o'! = o$) having the relationship p. Since I still use the essential idea of comparing common characteristics between the contexts for similar pairs of instances, the $y_{u,p}$ can be measured by the overall similarity between the pair u and all the pairs u' (equation 5.1), where U_p and $sim()$ is the similarity function comparing the contexts for two triples which will be introduced in Section 5.1.3.

$$y_{u,p} = \frac{1}{|U_p|} \sum_{u' \in U_p, u'! = u} sim(u', u) \tag{5.1}$$

5.1.1 Representing Context for Two Instances

Using the definition of Semantic Connection (Definition 12 on page 87), the context for a pair u is defined over a semantic connection space which is a vector space consisting of all possible semantic connections in the data set. Without ambiguity, we also refer the context representation as the representation for the pair itself which is shown below (equation 5.2), where n_{u,c_i} means the number of instantiations of the semantic connection $c_i(i \leq m)$ between the pair u, where m is the total number of distinct semantic connections in the data set.

$$V_u = [n_{u,c_1}, n_{u,c_2}, ..., n_{u,c_m}] \tag{5.2}$$

As semantic connections are based upon finite acyclic paths (Definition 2), the semantic connection vector space is also finite. But it would be impractical if no length limit is set on connections since very long connections may not convey a clearly meaning for relationship between two connected objects. According to observations in experiments of my previous work, in both the SWRC and DBpedia data sets, when the maximum semantic connection length (Definition 12) is five, most pairs of objects are connected by at least one connection. Also when the length increases from five to six, there is a diminishing returns in contextual information while the execution time increases rapidly. Therefore, I set the maximum length as five in this work. In the worst case, the vector space is $\mathrm{O}((2 * |R|)^5)$, where R is the set of predicates. But due to disjoint domains and ranges, not all predicates can follow another. For example, SWRC, which has about 100 predicates, has an actual semantic connection space of about 50, 000, which is much smaller than the theoretical worst case $(200)^5$. Clearly this context representation is applicable for computation on real world data, especially considering a sparse vector implementation can be used here.

5.1.2 Context Expansion

We note that a context with more semantic connections can usually supply more supporting evidence for the relation between the pair of objects. There are two factors that influence the number of semantic connections. First, the dataset is fully described over the provided vocabularies and so the instances in it have rich relational descriptions. Second, the semantic connection length limit can directly influence the number of connections. But these two conditions are neither general nor computationally efficient. First, considering that the open world assumption is applied to most Semantic Web data sets, some property values, i.e. relational descriptions, for an instance could be missing in the data. Second, recall that we set the length limit to five in this work, because of diminishing returns with larger max length. Third, some objects may still have few connections even in contexts with large length limits, due to the nature of the instances or the properties for them. So a better solution for expanding the context is needed.

Predicate co-occurence has been shown to correlate with instance similarity [75], so similar instances could give certain suggestions on predicate usage. Based on this idea, I proposed the following method to get more supporting evidence for a predicate usage between two instances. For each instance, I build a set of similar instances and call this set the expanded set. Because the semantic connections between two expanded sets are partially similar to the semantic connections between the original pair u, they can be viewed as partial semantic connections between the original pair.

The similarity used for building the expanded set is inspired by Semantic Web instance mapping research. Its general conditions are the same class type and a high percentage of the same predicate / object pairs. Specifically, the first condition is that the class type of a similar instance needs to be either the same as or the subclass of the class type of the original instance. Second, the predicates used on a similar instance need to be either the same as or the subproperties of the predicates used on original instance. Third, to find similar instances for the subject in a triple, I compare the predicate/object pairs; while for the object in a triple, I compare the predicate/subject pairs. The reason is that I want to determine the relationship between the original subject and object, so it is better to differentiate the instance similarity as a subject and as an object in a triple. When comparing object literal values, I use Jaro-Winkler [51] distance. Specifically, distance d_j of two given strings s_1 and s_2 is defined as $d_j = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right)$, where m is the number of matching characters and t is half the number of transpositions. Two characters from s_1 and s_2 respectively, are considered matching only if they are not farther than $\left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1$. Each character of s_1 is compared with all its matching characters in s_2. The number of matching (but different sequence order) characters divided by 2 defines the number of transpositions. For example. in comparing CRATE with TRACE, only ‘R’ ‘A’ ‘E’ are the matching characters, i.e, $m = 3$. Although ‘C’, ‘T’ appear in both strings, they are farther than 1.5, i.e., (5/2)-1=1.5. Therefore, t=0. In DwAyNE versus DuANE the matching letters are already in the same order D-A-N-E, so no transpositions are needed. In this work, if Jaro-Winkler distance between two strings is higher than 0.95, they are treated as the same. Finally to keep the expansion conservative and also make later computation efficient, the instances similarity threshold is set 0.8 in this work, i.e. 80% predicate and subject / object pairs used on the original instance are also used on the instances in the expanded set. Formally, the expanded set id defined as below.

Definition 16. *Given an RDF graph $G := (I, L, R, E)$ and an instance $i \in I$, the expanded set for this instance i used as subject is defined as*
$Expand\_sub(s) = \{x | \forall p, o_1, o_2, s.t. < s, p, o_1 > \in E, and\ < x, p, o_2 > \in E, and\ (o_1 = o_2\ or\ d_j(o_1, o_2) > 0.95,\ and$

$$\frac{|\{< s, p, o_1 >\}|}{|\{< s, p', o' > | < s, p', o' > \in E\}|} > 0.8\} \tag{5.3}$$

The above definition states as follows. Given an instance s, it returns a set of instance $\{x\}$. Each instance x is used as subject in set of triples $\{< x, p, o_2 >\}$ and there are another set of triples $\{< s, p, o_1 >\}$ which has s as subject and the same predicate. Meanwhile, o_1 and o_2 are identical or similar. Furthermore the percentage of such triples $< s, p, o_1 >$ are more 80% of all triples $\{< s, p', o' >\}$ in which s is used as subject. Using these two expanded sets for the pair of instances in a triple, I take all semantic connections between any pair of instances from these two sets as expanded context for the original triple.

If taking the two expanded sets as a new pair of instances $\bar{u}$, the context for $\bar{u}$

is represented the same as that for original pair u, except that the number of each connection is normalized by the multiplication of the size of two sets. Then the context for u is expanded as follows and renormalized by its magnitude.

$$\begin{aligned}\bar{V}_u &= (1-\alpha)[n_{u,c_1}, n_{u,c_2}, ..., n_{u,c_m}] + \alpha[n_{\bar{u},c_1}, n_{\bar{u},c_2}, ..., n_{\bar{u},c_m}] \\ &= [(1-\alpha)n_{u,c_1} + \alpha n_{\bar{u},c_1}, (1-\alpha)n_{u,c_2} + \alpha n_{\bar{u},c_2}, ..., (1-\alpha)n_{u,c_m} + \alpha n_{\bar{u},c_m}]\end{aligned}$$

The α is an expansion factor determines the significance level of the effect of the expansion. After simplifying the formula, we get a vector in the form of equation 5.2 as the general form of the representation for a pair of instances, since the above expanded form still is just a special form of it.

This context expansion enriches information for the pair of instances and also does not significantly increase the computation in later steps, specifically the number of distinct connections between each pair of objects in SWRC is only raised from 700 to 1000 on average after expansion. Given about 100 predicates in SWRC, the vector space has around 50k distinct semantic connections and each pair of objects have averagely 1k distinct connections after expansion. Thus the actual vector space is much smaller than the theoretical complete space which is 200^5 and each context needs far less space.

5.1.3 Semantic Similarity of Contexts

The cosine similarity is commonly used to compare two objects, especially when their features are represented in a vector space. However this measurement assumes that any two elements in the vector are independent and there is no similarity between them, which may not true in this problem domain. The similarity between different connections should affect the similarity between vectors. Considering similarity between elements in the vector space, I define the similarity between vectors as the sum of the similarities between all pairs of connections divided by the multiplication of the magnitude of two vectors (equation 5.4).

$$sim(u', u) = \frac{1}{||u||||u'||} \sum_{i=1}^{m} \sum_{j=1}^{m} n_{u,c_i} n_{u',c_j} s(c_i, c_j) \tag{5.4}$$

Because a semantic connection is a sequence of joined predicates which is formed under strong restrictions and conditions, I have the following considerations for the comparison between connections. First, the lengths of two comparable connections are equal because otherwise it will make it unnecessarily complex to determine which parts of connections are comparable. Second, to make two connections comparable, two properties at each corresponding position also need to be comparable, i.e. the pair of properties are equivalent or possibly similar. Third, partial matching between shorter connections usually gives a clearer semantics of a relation and should be given greater weight on final similarity. A multiplication of similarities on each segment on semantic connections can achieve this effect. Therefore the similarity between a pair of connections $s(c_i, c_j)$ is defined as the multiplication of similarity between every pair of properties at the corresponding positions (shown as follows),

$$\begin{aligned} s(c_i, c_j) &= s(< r_{i1}, r_{i2}, ..., r_{in} >, < r_{j1}, r_{j2}, ..., r_{jn} >) \\ &= \prod_{k=1}^{n} x_{ik,jk} \end{aligned} \tag{5.5}$$

where $x_{ik,jk}$ is the similarity between two properties r_{ik} and r_{jk}, if they are the same or equivalent, then it is 1, otherwise it is a real number in the range [0, 1). We will discuss how to calculate these values in section 5.2. The final form of $y_{u,p}$ is shown below.

$$y_{u,p} = \frac{1}{|U_p|} \sum_{u' \in U_p} \frac{1}{||u||||u'||} \sum_{i=1}^{m} \sum_{j=1}^{m} n_{u,c_i} n_{u',c_j} \prod_{k=1}^{n} x_{ik,jk} \tag{5.6}$$

Based on the research of property mapping on the Semantic Web, I think that two pre-conditions can indicate that two properties could be similar. First, two properties on two connections are used on the same direction. If they are used on opposite directions, we compare between the inverse property of one of them and the other. Second, two properties have a subsumption relation or have overlapping domains and ranges. Taking the SWRC data set as example, the vector space is around 50k and so all pair wise comparisons would be around 1.25 billion. However as mentioned above, any two instances in SWRC have roughly 1k semantic connections in the context and each semantic connection has just around 10 connections on average need to be compared, since other semantic connections that are theoretically similar also do not appear in the contexts to be compared with. Besides this advantage, because the pair wise property similarities $x_{ik,jk}$ are the model parameters, the conditions also limited them to not many. The initial value for the similarity of overlapping properties is the number of common pair of URIs divided by the number of total pairs; for the subsuming properties, it is the number of triples of subproperty divided by the number of all triples, including the inferred, of super property. After the learning model use these as parameters to optimize relationship classification on training data set, these parameters will be changed.

5.2 Learning Predicate Similarity

The context model for pairs of instances is used for comparing (dis)similarity between them and it consists of semantic connections which can be viewed as the hypothesized probabilistic rules. To determine how contexts are similar / different, these rules need to be associated with different weights to affect the comparison between two contexts. Furthermore these rules are connected through predicate usages and thus weights are finally affected by predicate similarities. Therefore the predicate similarities become the parameters to be learned for the classifier model. The following subsections will introduce how to learn the model parameters in order to determine the significance for each rule on discriminating the type of relations between instances.

5.2.1 Motivation

Maximum likelihood is a classical method for parameter estimation. The likelihood is the probability of the observed data as a function of the unknown parameters with respect to the current model, a specific one is element-wise loss on the triples estimated as follows, where $\hat{y}_{s,o,p}$ is the estimated score for the triple $< s, p, o >$, $y_{s,o,p}$ is a binary value depends on its state in the training data and $\hat{x}$ is the vector of predicate similarities, i.e. parameters.

$$\arg\min_{\hat{x}} \sum_{(s,p,o)\in I\times R\times I} (\hat{y}_{s,p,o} - y_{s,p,o})^2 \tag{5.7}$$

But this estimation has several problems. First it uses the 0/1 interpretation scheme. In other words, it encodes each positive sample as 1 and interprets the remaining data as 0. Considering open world, it would be false to think pairs of instances which are not used as the subject and object or vice versa in any triple have no appropriate predicate from the ontologies that the data conform to to represent their relation. Second, it assumes the data set to be dense. If all elements that are not in it are assumed to be 0, even for a small dataset like SWRC, the 0 values dominate the 1 by many orders of magnitude. Specifically, if the sparsity is defined as $1-|E|/(|I|*|R|*|I|)$, the sparsity of SWRC is over 99.9%, given 73k for $|E|$, 31k for $|I|$ and 123 for $|R|$. Third, as the system is trying to give a probabilistic suggestion, trying to fit to the numerical values of 1 and 0 is an unnecessary constraint. Instead only the qualitative difference between a positive and negative example is important. That means it would be good enough if $\hat{y}$ of a positive example is sufficiently larger than that of a negative example. Fourth, maximizing likelihood could result in overfitting and its computation is globally.

5.2.2 Learning Model for Predicate Similarity

Another type of parameter estimation is to calculate the gradient descent iteratively and locally. The model I used is modified from its application in the tag prediction problem [99]. First, I give a brief introduction of tag prediction or tag recommendation from a perspective related to relationship classification in this problem.

Tagging, in general, allows users to describe an item (e.g. website, song, friend, etc.) with a list of words (i.e. tags). Tags can be used e.g. for organizing, browsing and searching. Tagging is a popular feature of many websites like last.fm, delicious, facebook, flickr. With tag recommendation a website can simplify the tagging process for a user by recommending tags that the user might want to give for an item. As different users tend to give different tags for the same item, it is important to personalize the recommended tags for an individual user. That means the tag recommender should infer from the already given tags, which tags a certain user is likely to give for a specific item. For predicting a personalized list of tags for an item, the tag recommender should use the tagging behaviour of the past of this and other users as well as the tags for this and other items. Interesting about tagging data is that it forms a ternary relation between users, items and tags. This makes it different from typical recommender systems where the relation is usually binary between users and items. However it is similar to a ternary relation between subjects, objects and predicates on the Semantic Web. Exploiting all information of the ternary relation is a key in tag recommendation. A second problem that many tag recommendation try to solve is the data interpretation as usually only positive feedback is present in a tagging system. In my research problem, there is a training data set that is assumed generally correct without labels. Thus, similarly, the data set are usually only the positives. However this is on a different problem domain with different background, scenario and parameters. There are a number of challenges to adapt the model from tag prediction to relationship classification in Semantic Web data.

For a pair u of instances, the algorithm ranks predicates by $y_{u,p}$ (equation 5.6), which is a scoring function for this pair u and a given predicate p and contains some parameters, i.e. predicate similarities. The objective function (equation 5.8) maximizes the ranking statistic AUC (area under the ROC-curve). An ROC curve is a graphical representation of the trade off between the false negative and false positive rates for every possible cut off. When the AUC is maximum, the system achieves idea balance between false negative and false positive.

$$AUC(\bar{x}, u) = \frac{1}{|P_u^+||P_u^-|} \sum_{p^+ \in P_u^+} \sum_{p^- \in P_u^-} h(y_{u,p^+} - y_{u,p^-}) \tag{5.8}$$

To make the AUC differentiable, I used the s-shaped logistic function $h(x)$ which can make the learning curve exhibit a progression from small beginnings that accelerates and approaches a climax over iterations.

$$h(t) = \frac{1}{1 + e^{-t}} \tag{5.9}$$

Then using gradient descent, AUC has to be differentiated with respect to all model parameters. For each pair $u \in U$, where U is all possible pair of instances in the data, P_u^+ is the set of predicates that are already used between the pair u in the data while P_u^- is the set of predicates that are not used between the pair u in the data. The overall optimization task with respect to the ranking statistic AUC and the observed data is then:

$$\arg\max_{\bar{x}} \sum_{u \in Ps} AUC(\bar{x}, u) \tag{5.10}$$

With this optimization (1) missing values are taken into account because the maximization is only done on the observed pairs U and (2) the model is optimized for ranking. In all, this criterion takes into account all motivations discussed in section 5.2.1. The model parameters $\mathbf{x}$ which is a vector of all possible pairs of predicate similarity introduced in Section 5.1.3 are updated:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{x}} & AUC(\bar{x}, u) \\ = & \frac{\partial}{\partial \mathbf{x}} \frac{1}{|P_u^+||P_u^-|} \sum_{p^+ \in P_u^+} \sum_{p^- \in P_u^-} h(y_{u,p^+} - y_{u,p^-}) \\ = & z \sum_{p^+ \in P_u^+} \sum_{p^- \in P_u^-} t_{p^+,p^-} \frac{\partial}{\partial \mathbf{x}} (y_{u,p^+} - y_{u,p^-}) \end{aligned}$$

with:

$$\begin{aligned} t_{p^+,p^-} &= h(y_{u,p^+} - y_{u,p^-})(1 - h(y_{u,p^+} - y_{u,p^-})) \\ z &= 1/|P_u^+||P_u^-| \end{aligned}$$

and

$$\begin{gathered} \tfrac{\partial}{\partial \mathbf{x}}(y_{u,p^+} - y_{u,p^-}) \\ = \tfrac{\partial}{\partial \mathbf{x}}\big(\tfrac{1}{|U_{p^+}|} \textstyle\sum_{u' \in U_{p^+}} \tfrac{1}{||u||||u'||} \sum_{i=1}^m \sum_{j=i}^m n_{u,c_i} n_{u',c_j} \prod_{k=1}^n x_{ik,jk} \\ - \tfrac{1}{|U_{p^-}|} \textstyle\sum_{u' \in U_{p^-}} \tfrac{1}{||u||||u'||} \sum_{i=1}^m \sum_{j=i}^m n_{u,c_i} n_{u',c_j} \prod_{k=1}^n x_{ik,jk}\big) \end{gathered}$$

$$\begin{gathered} = \tfrac{1}{|U_{p^+}|} \textstyle\sum_{u' \in U_{p^+}} \tfrac{1}{||u||||u'||} \sum_{i=1}^m \sum_{j=i}^m n_{u,c_i} n_{u',c_j} \tfrac{\partial}{\partial \mathbf{x}} \prod_{k=1}^n x_{ik,jk} \\ - \tfrac{1}{|U_{p^-}|} \textstyle\sum_{u' \in U_{p^-}} \tfrac{1}{||u||||u'||} \sum_{i=1}^m \sum_{j=i}^m n_{u,c_i} n_{u',c_j} \tfrac{\partial}{\partial \mathbf{x}} \prod_{k=1}^n x_{ik,jk} \end{gathered}$$

I noted that this equation contains a lot of computations that can be reused for each round, e.g. the derivative of the similarity between two connections are not changed within each iteration. So I use some memoization techniques to eliminate many repeated computations and update the memoized table once after each iteration. Thus for each pair $u \in P_s$, the model parameters $\mathbf{x}$ are updated as follows:

$$\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \tau \cdot \frac{\partial AUC}{\partial \mathbf{x}} \tag{5.11}$$

where τ is the learning rate which I set as 0.05. This equation means after the model learns from each observed triple to increase the gap between the positives and the negatives, it updates the model parameters, i.e. predicate similarities, based on the learning rate.

5.2.3 Dimensionality Reduction for Learning

The purpose of the learning is to find the common characteristics underlying different triples with the same predicate. Meanwhile these characteristics can maximize the gap between the triples of this predicate and the triples of other non-equivalent predicates. Ideally, we should avoid the unnecessary computation that may not improve or change the model parameters enough to affect the final results above a certain threshold.

I found that the significant changes of model parameters during the learning process are incurred by only some of new positive or negative examples. These examples usually have a vector representation different enough to the previous ones that are already examined by the learning model. Thus if the next example is quite similar to any previous one that is already computed, it is expected that the changes of model parameters usually would not be significant. To save this type of computation, I proposed to cluster these similar examples and then treat each cluster as a single training sample. The idea is similar to k-means clustering, which is NP hard and needs many iterations even in approximate optimal algorithms. Since the requirement on the learning time is not as critical as running time, to keep this optimization conservative, the criterion of the clustering is that the cosine similarity between any two samples in the cluster is above a certain threshold γ. The higher γ is, the more/smaller clusters are. In the clustering process, for triples of each predicate, the system first computes and ranks the similarity of all pairs of them. Next the system repeats the following process until the similarity of the pair on top of the list is below the threshold γ. If both triples in the pair are not clustered, the system creates a new cluster consisting of these two. If one of them is already clustered while the other is not, then the other one would be merged into this cluster as long as its similarity with at least one in the cluster is higher than γ. Or if both are clustered and the similarity of any pair of triples from these two clusters is above the threshold γ, these two clusters are merged as a new one. Finally, the system averages each cluster into a single sample by averaging the vector representations of triple in this cluster. This process essentially condenses the training set.

5.3 Experiments

In the experiments for this algorithm, the training and test data set are the same as what I chose for the previous algorithm (Section 4.6), i.e. SWRC and DBpedia. To improve the connectivity on the RDF graph, I treat some literals as resources that mainly are object values of some inverse functional datatype properties and treat them as common URI resources. Inverse functional means that the subjects of these reused literals are indicating the same real world object. Through this, they play the roles as equivalent objects on our enhanced RDF graph. The properties are *email*, *homepage*, *title*, *url*, *isbn*. These properties usually can be determined by the ontology, however they also can be manually input.

After the data set is split into test and training set, the experiment process generally is as follows and according to testing requirement on each specific aspect of the system, some of the steps might be modified or removed. First, the system builds contexts for training samples. Second, the system clusters the training samples (introduced in section 5.2.3) in order to improve the learning time. Third, the system learns model parameters (predicate similarities) on training samples, given their initial values. Fourth, I input testing triples (1000 positives and 1000 negatives) with the condition that the predicates of these testing triples are unknown to the system, both the positives and the negatives. Thus system gives a top ranked predicate by according to the scoring function 5.6 which uses the learned parameters. Finally I check the resulting predicate that is determined by the relational classifier with highest score. For positive samples, the system is expected to entail the correct predicate, which means the system can detect the abnormality if the predicate is not used between the pair of instances. Note, by making the correct predicates the positive samples, precision is the number of times we are right when we think the predicate is the correct one. This is different from precision in the real error detection experiments: where precision is the number of reported anomalies that are actually errors. For negative samples, it is expected that no relation between the objects is entailed by the system with a score above the threshold β and so the system reports it as no credible relation. Following the typical assessment approach for classification problems, all experimental results are measured in terms of precision, recall and F-measure. Specifically, given equal amount of positives and negatives in test set, I get the confusion matrix for them and calculate the precision as true positives / (true positives + false positives) and recall as true positives / (true positives + false negatives).

5.3.1 Parameters Analysis

After the learning process, I first checked the result of model parameters, i.e. predicate similarities (shown in table 5.1). For each pair of predicates, the initial value is the percentage of their actual overlapping usages in the data set. For example 0.53 is the initial value of the similarity between *hasTopic* and *topic*, which means that there are about 53% of pairs of objects in these triples used both the predicate *hasTopic* and *topic*. In reality, these initial values can be input by the user, if the user believe a pair of properties are similar. The result value is the predicate similarity given by the system after the learning process. For example, 0.94 is the similarity between the predicate *hasTopic* and *topic* computed by the system. Since

Table 5.1: Some predicate pairs from SWRC and DBpedia and their result similarity values.

Predicate Pair	Initial / Result
http://data.semanticweb.org/ns/swc/ontology/hasTopic, http://xmlns.com/foaf/0.1/topic	0.53/0.94
http://purl.org/dc/terms/creator^{-}, http://xmlns.com/foaf/0.1/made	0.76/0.91
http://dbpedia.org/ontology/SpaceMission/nextMission, http://dbpedia.org/ontology/SpaceMission/previousMission^{-}	0.91/0.98
http://purl.org/dc/terms/creator, http://xmlns.com/foaf/0.1/maker	0.87/0.93
http://swrc.ontoware.org/ontology/author, http://purl.org/dc/terms/creator	0.69/0.91
http://dbpedia.org/ontology/Work/subsequentWork, http://dbpedia.org/ontology/Work/previousWork^{-}	0.50/0.86
http://swrc.ontoware.org/ontology/author, http://xmlns.com/foaf/0.1/maker	0.71/0.92
http://swrc.ontoware.org/ontology/title, http://purl.org/dc/terms/title	0.20/0.82
http://dbpedia.org/ontology/Person/predecessor, http://dbpedia.org/ontology/Person/successor^{-}	0.43/0.83
http://swrc.ontoware.org/ontology/author^{-}, http://xmlns.com/foaf/0.1/made	0.80/0.92
http://xmlns.com/foaf/0.1/accountName, http://rdfs.org/sioc/ns/name	0.25/0.73
http://dbpedia.org/ontology/Organisation/foundationPerson, http://dbpedia.org/ontology/Organisation/foundationOrg.	0.24/0.67
http://swrc.ontoware.org/ontology/affiliation^{-}, http://xmlns.com/foaf/0.1/member	0.76/0.91
http://xmlns.com/foaf/0.1/homepage, http://xmlns.com/foaf/0.1/page	0.08/0.51
http://dbpedia.org/ontology/Person/nationality, http://dbpedia.org/ontology/Person/birthPlace	0.02/0.43

I do not have ground truth of the similarity, I manually checked 20 pairs of them. Most computed similarity values are higher than the initial ones (it can be noted in the table). The reason is that if some actual overlapping usages appear in the data set, they indeed have some similarity and so make the authors of the data set free to choose either of the predicates at least for some triples. Second, usually the authors of a data set would not create triples using both of the predicates at most cases, since the data might be simply redundant. Combining the above two reasons, the initial value, i.e. the percentage of actual overlapping usages usually is far less than their similarity measure and is just a hint of possible similarity in semantics. I also noticed that two of the computed values are lower than the initial value, i.e. actual overlapping percentages. For example, the predicate pair of *instrument* and *occupation* have about 4% actual overlapping and the result value given by the system is lower than 0.01. This example again shows that the overlapping is just a hint and the hint may be either true or false. In this case, it is a bit arguable that whether these two predicates are similar, because it might depend on the domain the data set focus on. But in DBpedia, given the domain of *instrument* is $owl : Thing$, i.e. anything in the domain that DBpedia covers, I think the system results truly reflect their relationship, i.e. not much similar. In general, I note that the results are consistent with my expectations on these predicate similarities, although no mappings are defined between them. They include possible equivalent property pairs (e.g. d:hasTopic and f:topic), subsumption property pairs (e.g. f:homepage, f:page), datatype property pairs (e.g. s:title and p:title), possible inverse property pairs (e.g. ds:nextMission and ds:previousMission). Note these intermediate results could also be used for ontology alignment.

5.3.2 Results

In the first experiment, I compared the F-score, precision and recall when the expansion factor α (Section 5.1.2) and the threshold β (Section 5.3) varies (shown in Figure 5.2 and Figure 5.3). From the results, we can see that the system without

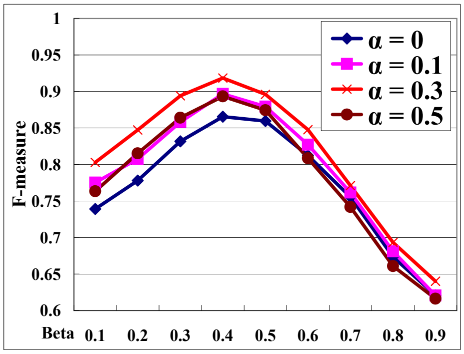


Figure 5.2: The effect of different expansion factor α and different credible relation threshold β on F-score and SWRC.

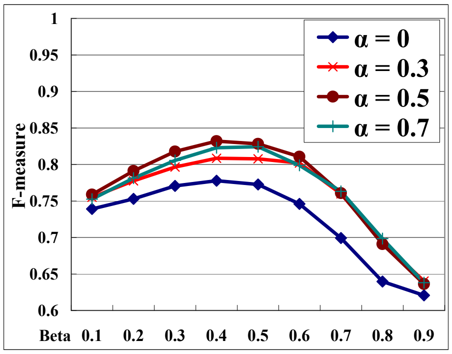


Figure 5.3: The effect of different expansion factor α and different credible relation threshold β on F-score and DBpedia.

expansion ($\alpha = 0$) basically is worse than any systems with expansion and when using the contexts with expansion, the systems with $\alpha = 0.3$ and $\alpha = 0.5$ are the best on SWRC and DBPedia, respectively (in following experiments, the α will be set as these values in two data sets respectively). When the system does no expansion, it is similar to my previous system and the performance gaps between this system and that one are in range of 1% to 5% on different points of the curve. To not overwhelm readers, the lines with other alpha values are not shown here. The reason DBpedia needs more context expansion is that it has less relational descriptions for instances than SWRC. For β, the system performs the best on both data sets when it is 0.4.

In the second experiment, I compared the system performance with and without predicate similarity learning when the number of missing triples in the data set varies (Figure 5.8). As introduced, the predicate similarity is the parameters for the classifier determining the type of relationship. In the system without learning, the property similarities are set as the percentage of their actual overlapping usages in the data set. While in the system with learning, the property similarities are set as the learned results. In the test, we randomly removed some portion of triples in the data set in order to simulate the open world assumption. From the results, we see the system with learning is better, since the performance gap between two systems become bigger when more triples in the data set are missing. The reason is that as

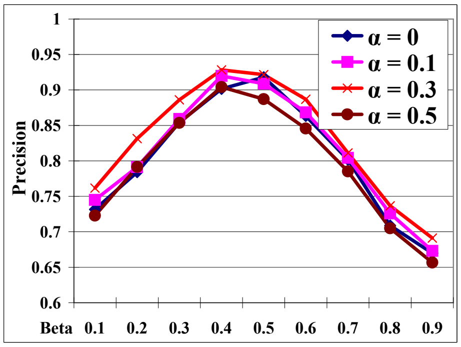


Figure 5.4: The effect of different expansion factor α and different credible relation threshold β on precision and SWRC.

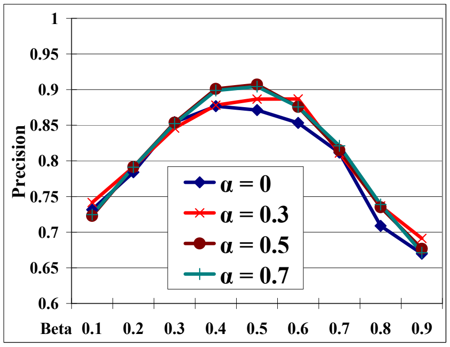


Figure 5.5: The effect of different expansion factor α and different credible relation threshold β on precision and DBpedia.

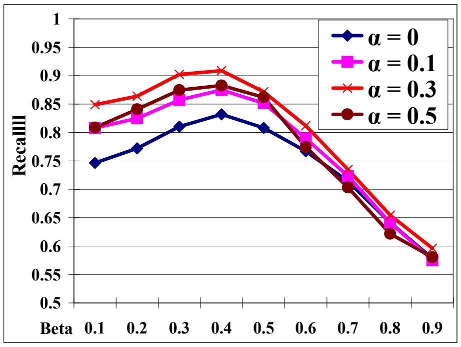


Figure 5.6: The effect of different expansion factor α and different credible relation threshold β on recall and SWRC.

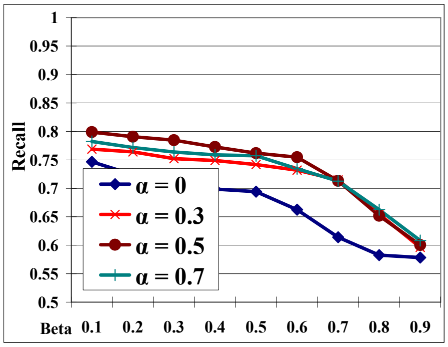


Figure 5.7: The effect of different expansion factor α and different credible relation threshold β on recall and DBpedia.

the context information shrinks, the exact matching for semantic connections will be rarer. Then partial matching for semantic connections affect the result more and so the effects of predicate similarity become larger.

The next experiment is to test various sampling thresholds γ (Section 5.2.3) on SWRC. In this experiment (shown in Figure 5.9), time is measured by wall clock and the program is run with one thread on a Sun workstation with 8 Xeon 2.93G cores. When sampling, the time includes both clustering and learning. We note that when $\gamma = 0.85$, on average every two samples are compressed into one because the learning time is reduced to about 1/4 of original learning time. For this value,

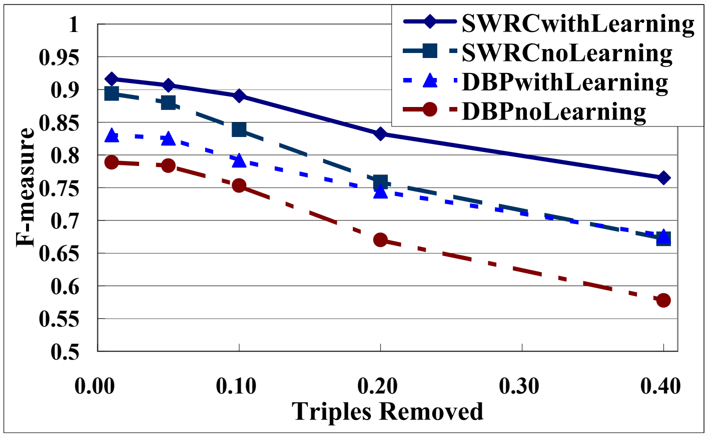


Figure 5.8: The effect of clustering threshold γ on learning time and performance.

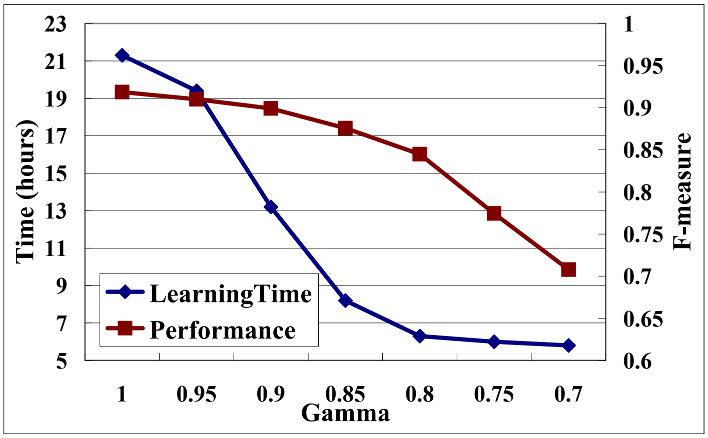


Figure 5.9: The effect of clustering threshold γ on learning time and performance.

the F-measure only decreases by 4%. But when $\gamma < 0.8$, the time saved is less and performance comparatively drops faster. In other words, we see diminishing returns: the computation/F-measure trade off becomes unfavorite.

The last group of experiments is designed to compare if the system is better than my previous system on two aspects. The one aspect is about closed world assumption or open world assumption, which is different in the two systems. The the other aspect that I want to compare is about the assumption that the training data set is generally correct. Fig. 5.10 and 5.11 shows these comparisons. The lines of SWRC_closed and DBP_closed are the performance of my previous system.

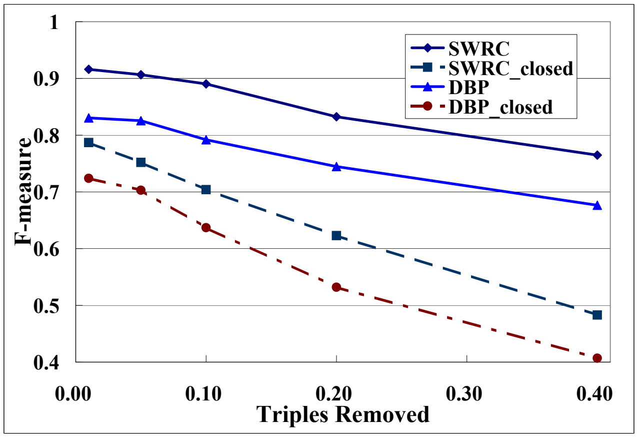


Figure 5.10: Comparison of the effects of removing data from the referenced data set on the current system and on the previous system.

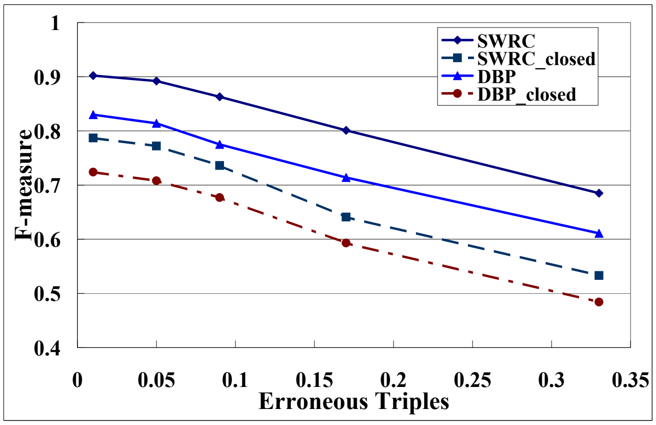


Figure 5.11: Comparison of the effects of erroneous data in the referenced data set on the current system and on the previous system.

The first clear observation is that, as expected, this system performs much better than the previous system on every points in the two figures. Specifically, when 10% triples removed from training set, the system performance gaps increase by 2% on both data sets. The second observation is that this system is affected much less than the previous system when the open world assumption is applied to the data set, i.e. when significant portion of data is removed, because the slope of performance drop on this system is not as sharp as that of the previous system. Fig. 5.11 shows experiments based on training data sets injected with some portion of synthetic incorrect triples. Since the system still assumes that the training set is generally correct, it is expected that the more incorrect triples injected, the worse performance is. It can be seen when about 10% of triples are incorrect, the F-measure of both systems only drops about 3%. The reason the system is robust is that the probabilistic rules are learned based on the agreement between data, and small portion of incorrect triples is hard to make many other data agree with it. But it can also be recognized that, when the data set has some incorrect data, the trends of two system performances are similar.

In summary, the major observations in these experiments are as follows. The experiments basically show that the system satisfied the two design goals: 1) in a Semantic Web data set to which the open world assumption is applied, the system needs to discover common characteristics that are similar to those used in the previous system; 2) when computing the similarity between contexts of triples, the system needs to consider the open world assumption. In addition, the predicate similarities, i.e. the classifier parameters, as the result of learning, are also useful for other tasks, e.g. ontology alignment. However in the experiments, it is also can be noticed that the system accuracy is affected more when the training data set is not generally correct, than when other aspects are changed, e.g. removal of triples. I believe it might not be a usual case that users take a data set that is already recognized as not generally correct for training purpose. But I can still imagine some cases when there is no clean data set is available for training in advance. Therefore in those cases, it would be necessary to design another system dealing with this case.

Chapter 6

Data Correctness without Training

The previous two chapters addressed some research problems of detecting quality issues in Semantic Web data. More importantly, they clearly demonstrated the essential forms of useful, potential patterns in many Semantic Web data. However as I summarized through experiments on previous systems, it can be recognized that both of these algorithms require a training data set which is assumed to be generally correct (less than 5% errors in a rough estimation). Although this general idea that training on clean data and then applying the system to new data is frequently used in various data quality techniques based on learning or classification [29], there are many real world cases where there is no clean data set is available for training in advance. Therefore it would be necessary to design a system for dealing with these cases. Furthermore, the systems I have designed in previous chapters and to be designed in this chapter are all unsupervised learning systems. Thus, from a machine learning theory perspective, learning from noisy data is also an important scenario to be considered [29]. So the primary research question for this scenario would be whether the system can without supervision discover correct patterns in the data to be evaluated rather than learning from training data in advance. Another research problem or challenge is that the data where patterns are discovered is expected to contain significant incorrect data, instead of being assumed to be generally correct as the training data in previous systems. There is no need to alter the form of the learned patterns. But the discover process may need to be more complex in order to take into account the data with different reliability and then differentiate reliability of the patterns. The general process of a system that overcomes these challenges is to extract patterns from data, compute/refine probabilities of patterns and in turn use patterns to evaluate data. An iterative approach appears to be suitable for this problem [103].

6.1 Approach Overview

Based on the research problem posed and the brief analysis given above, now I give a overview of the designed iterative approach (shown in Fig. 6.1). Compared with algorithms in Chapter 4 and 5, the first major change is to consider the possibility of noise in the data when learning. Recall from section 4.5 that patterns are based on semantic connections and instantiations of semantic connections affect the weights of patterns. Instantiations for a semantic connection are Paths (Definition 12) on an RDF graph on which the semantic connection is embedded as a sequence of predicate usages. The patterns to be discovered here can be similar in form, i.e. they might be still based on semantic connections and affected by instantiations of this pattern, i.e. some Paths (Definition 2) on the RDF graph. However, if the confidence in the underlying triples that makes up two Paths is different, then the weight of these Paths should be different, whereas previously, I have treated all Paths as equal evidence for semantic connections. Besides the major change for system accuracy, another major change is for system efficiency. Because the system will retrieve context for every triple in each iteration, the context extraction can be changed from a per triple basis to per RDF graph basis. There are three reasons for that. First I realized that many contexts for triples have overlapping triples. Thus constructing contexts on a per triple basis results in many redundant computations. Second, semantic connections are extracted from Paths in the contextual RDF graph for triples. A semantic connection is supporting evidence for a triple only if a Path where this semantic connection is extracted shares the same instances on the two ends with this triple on the RDF graph. Thus these Paths and the triple actually forms cycles on the undirected RDF graph. Then for each cycle, given any segment, Paths based on the remaining segments could be an evidence that the system is looking for. Therefore, searching the same cycle starting at different points on it is redundant when the system tries to search supporting evidence for different segments in the cycle. Third, since the weight of a pattern will be computed based on a triple's truth probability, we need an efficient mechanism to retrieve the triples that compose of instantiations of any pattern repeatedly during the iterative learning process. Although, these resulting triples can be stored and so can be reused in later iterations, considering scalability, the space complexity of the storage would be in proportion of data set size and much likely huge (almost quadratic, since lots of overlapped triples that are composed of instantiations of patterns). Therefore, I need to invent an efficient way to retrieve triples for a given pattern and a more efficient data structure is necessary.

The detailed main process of the approach is in Algorithm 2. First, every triple is assigned a prior truth probability. The function $Prior\_Prob$ in Algorithm 2 returns P_E which is a vector of triple's probabilities corresponding to each triple in E. A straightforward assignment without prior knowledge would be using a uniform distribution, i.e. all triples are equal likely (in)correct, e.g. 0.5. The other type of meaningful assignment is to take into account the prior or domain knowledge of the data. An example prior probability assignment based on prior knowledge could be a function of the source that contains the triple, e.g. all triples in a certain source have a particular value. Next (see Section 6.2.1), the system builds a summary graph over the original RDF graph in order to efficiently extract all candidate semantic dependencies (SD) and their instantiations (function $Find\_SD()$ in Section 6.2.2). A semantic dependency is a rule that is similar to association rules which have been utilized in some previous research work on data cleansing and outlier detection (as discussed in Section 2.2.3). Using the instantiations and prior probability of triples, the system determines the truth probability of each SD. The function $Get\_Prob\_SD()$ (see Section 6.3.1) returns P_S which is a set of SD/probability pairs. Taking a logic perspective, these SDs are also probabilistic axioms over the original ontology. The next important step (function $Refine\_Prob\_SD()$ in Section 6.3.2)

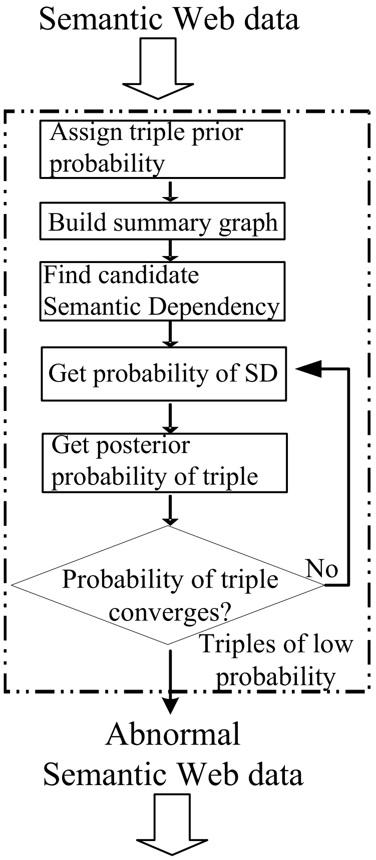

Semantic Web data
Assign triple prior probability
Build summary graph
Find candidate Semantic Dependency
Get probability of SD
Get posterior probability of triple
Probability of triple converges?
No
Triples of low probability
Abnormal Semantic Web data

Figure 6.1: The work flow of the system without referenced data set.

is to refine the beliefs of them according to their logical relations, e.g. inconsistencies. The final step in each iteration (function *Get_Prob_Triple*() in Section 6.3.3) is to get a score for each triple based on the number and the probability of the SDs that could corroborate it. Then the system transforms the triple score into a prior probability to be used in the next iteration until the stopping condition is satisfied.

Shown in the algorithm, the stopping condition of the iterative process is when the difference between the posterior probability and prior probability of every triple is less than a predetermined threshold. The iterative process would terminate under all the three possible situations about the change of a triple's score. First, the score may be monotonic, either increasing or decreasing. Since the triple's score will be bounded within [0, 1] (details will be given in section 6.3.3), in the extreme case, a triple finally will get 0 or 1 and there is no change any more. Thus it will satisfy the stopping condition. Second, the score may not be monotonic but the absolute value of the score's gradient decreases. This will satisfy the stopping condition at certain time when the change falls into a given threshold. The last case is that the triple score is oscillating and the changes do not decrease. It means that the system can only bound the triple score into a range. Due to the weak law of large numbers, if we take the average over all iterations as its score, then the score is expected to be close to the expected value. The weak law of large numbers states that the sample average converges in probability towards the expected value $\overline{X}_n \xrightarrow{p} \mu$ when $n \to \infty$ (see Probability Theory [62] Chapter 1). That is to say that for any positive number ϵ, $\lim_{n\to\infty} \Pr\big(\,|\overline{X}_n - \mu| > \varepsilon\,\big) = 0$. Interpreting this result, the weak law essentially states that for any nonzero margin specified, no matter how small, with a sufficiently large sample there will be a very high probability that the average of the observations will be close to the expected value, that is, within the margin. Due to this reason, for every triple, the algorithm keeps its average probability until the current iteration and compare it with the average probability in previous iteration, when the first two stopping conditions do not apply.

6.2 Semantic Dependency

I first define semantic dependencies, compare them with association rules and then introduce how to efficiently find them in a given RDF data set.

Definition 17. *Given an RDF graph G:=(I,L,R,E), a Semantic Dependency (SD) s in graph G is* $x \to y$*, where* x *is a semantic connection,* $y \in R$ *and* $\exists p \in Inst(x, G)$*,* $First(p) = I_0$ *and* $Last(p) = I_n$*, s.t.* $(I_0, y, I_n) \in E$*. For convenience, let* $LHS(s) = x$ *and* $RHS(s) = y$*. A semantic dependency has an associated probability.*

Recall association rules as discussed in chapter 2, the data set in question is defined as the basket data $B = \{b_1, b_2, ..., b_n\}$, where each basket $b_i \subseteq I$ is a collection of items, and where $I = \{i_1, i_2, ..., i_k\}$ is a set of k elements. Then the general definition of an association rule is $A \Rightarrow B$, where A and B are disjoint sets of items,

Algorithm 2 Main(G, α), G = (I, L, E, R) is an RDF graph, α is a stopping threshold

```
$P_E \leftarrow Prior\_Prob(E)$ //$P_E$ is a vector of triple's probabilities
$Avg_E \leftarrow P_E$
$G' \leftarrow Build\_Summary(G)$ //Build summary graph G'
$SD \leftarrow Find\_SD(G', G)$
$i = 1$
for $i < \infty$ do
    $P'_E \leftarrow P_E$
    $Avg'_E \leftarrow Avg_E$
    $P_S \leftarrow Get\_Prob\_SD(SD, P'_E, G)$
    $P_S \leftarrow Refine\_Prob\_SD(P_S)$
    $P_E \leftarrow Get\_Prob\_Triple(P_S, G)$
    $Avg_E \leftarrow (P_E + i \cdot Avg'_E)/(i+1)$
    if $\forall t \in E, |P'_E[t] - P_E[t]| < \alpha$ or $|Avg'_E[t] - Avg_E[t]| < \alpha$ then
        break
    $i \leftarrow i + 1$
```

where A is referred to as the antecedent (or left-hand side) of the rule, and B as the consequence (or right-hand side) of the rule. A natural language version of a real association rule is phrased as: "50% of people who buy both diapers and potatoes, also buy beer."

Comparing between semantic dependency and association rules, they both consist of two parts: left-hand side (LHS) and right-hand side (RHS); they both have similar implication semantics: the RHS is conclusion of LHS (or premise). So I compare each side respectively. On the left side, an association rule has a set of items. The set of items just means they co-occurred in some baskets, which in this thesis could be considered contexts. In other words, association rules typically do not consider the order of items either within a basket or across baskets. In a semantic dependency, the left-hand side is also showing a co-occurrence of some items, i.e. predicates. However semantic connections (Definition 12) are defined as an ordering over predicates. This order gives more semantics, because they are connected by certain join conditions. In addition, this characteristic also serves as important heuristic to be used in the process for discovering these rules. The right-hand side of an association rule is another item which co-occurs with the set of items on the left-hand side. Similarly, the right-hand side of a semantic dependency also co-occurs with the set of left-hand side predicates. However, the co-occurrence between the LHS and RHS of a semantic dependency has an additional restriction that they share the same instances on the two ends. After comparing on two sides each, taking association rule as a whole, it has confidence and support measure. The semantic dependency has also a probability measure which is similar to the confidence. But we do not use support for any given SD, we use these SDs through majority voting for a triple. These differences are all designed to capture the characteristics of RDF data which is a graph data model and support semantics specified entailment, since every semantic dependency can be represented as axioms using description logic (DL) using OWL 2 features[1] (as discussed in Section 2). For example a SD $locatedIn \circ partOf \rightarrow locatedIn$ can be represented as $locatedIn \circ partOf \sqsubseteq locatedIn$, which means that if x is located in y and y is part of z then x is located in z.

6.2.1 Summary Graph

As I discussed earlier, the system requires an efficient data structure to discover SDs and support the retrieval of triples that are instantiations of SDs. The definition of SD shows that the LHS of a SD is a semantic connection. So the precondition of discovering SDs is to find all semantic connections that can contribute to SDs first. A semantic connection is extracted from Paths on the RDF graph. Each pair of consecutive predicates on a semantic connection is related because some triples of the preceding predicate share the objects with triples of the subsequent predicate. Besides the LHS of a SD is based on predicate composition, both ends of the RHS of a SD is also joined with both ends of the LHS of this SD. An intuitive way to find such join connections is to compute intersections between all pair wise property instances. But enumerating all possible semantic connections by computing

[1] http://www.w3.org/TR/owl2-overview/

the instance intersections between all pair wise relations is very time consuming, especially considering most of the pairs do not have intersections. Thus a goal is to reduce the number of joins required to discover a semantic connection. Based on this observation, I want to first prune those pairs of sets that cannot have overlap.

Since finding instantiations of these semantic connections is similar to using the Basic Graph Pattern (BGP) in a SPARQL query [89] and the pruning to be designed is similar to BGP query optimization, my solution for improving efficiency of discovering SDs is inspired by the BGP query optimization. Query optimization is a fundamental and crucial subtask of query execution in database management systems and likewise in RDF triple store management systems. One of important optimization techniques for an RDF triple store management systems is static query optimization, i.e. a join order optimization of triple patterns performed before query evaluation. The optimization goal is to find the execution plan which is expected to return the result set fastest without actually executing the query or subparts. This is typically solved by means of heuristics and summaries for statistics about the data. The execution time of queries is heavily influenced by the number of joins necessary to find the results of the query. Therefore, the goal of query optimization is (among other things) to reduce the number of joins required to evaluate a query.

A single computation on the intersection of two relations $p1$ and $p2$ is like a query on instances that can be defined as the SPARQL query below.

```
SELECT ?x
WHERE {
 ?a p1 ?x
 ?x p2 ?b
}
```

To make such existential queries be more efficient, I built a summary graph $G' = (I', L', E', R)$ corresponding to the original graph $G = (I, L, E, R)$. This summary graph rules out most of the pairs of predicates that have no join relations in the data. Each connected pair of edges on the summary graph means that it is possible that the two predicates represented by the two edges can have join relation. An individual in I' represents individuals in I which are used as the domains or ranges of same group of properties. Formally graph G' is a summary of an RDF graph G if there is a mapping function $f : I \rightarrow I'$ that satisfies the following constraints:

1. if $P(a, b) \in E$ then $P(f(a), f(b)) \in E'$.
2. if $P(a', b') \in E'$ then $\exists a, b \in E$, s.t. $a' = f(a), b' = f(b)$ and $P(a, b) \in E$.

The process for constructing the summary graph (the function *build_Summary()* in the Algorithm 2) can be done efficiently from G using SPARQL queries. The process to construct the summary graph compresses one predicate on each iteration by merging all the domain objects and range objects into two summary representative nodes. Meanwhile it merges all other property links connected with the nodes being merged. When all predicates are compressed, the iteration stops. In the example summary building process (shown in Fig. 6.2), Fig. 6.2 (b) is an intermediate state that the node *P1* and *P2* are merged because they are both used as the object value of the predicate *author* and the node $A1$ and $A2$ are merged because they are both used as subject of the predicate *author*. Meanwhile, all other edges from and to these nodes are merged, e.g. the edge $hasTopic$ from node $A1$ is changed to starting from new node $A12$. If two edges are connected through a common node on the summary graph, the implication is that the two properties represented by them possibly can be joined. Thus this summary graph facilitates the process of finding all candidate semantic connections and thereby SDs as well, since semantic connections are the basis of SDs. Generally, the summary graph G' is dramatically smaller than the original RDF graph G and is easily implemented as in-memory graph model.

6.2.2 Finding Candidate Semantic Dependencies

In this and following subsection, I discuss the two main steps in the function *Find_SD()* in Algorithm 2. As defined, one of the important pre-conditions of a semantic dependency is that the first instance of some instantiations of premise is the same as the subject of instantiation of the conclusion and the last instance

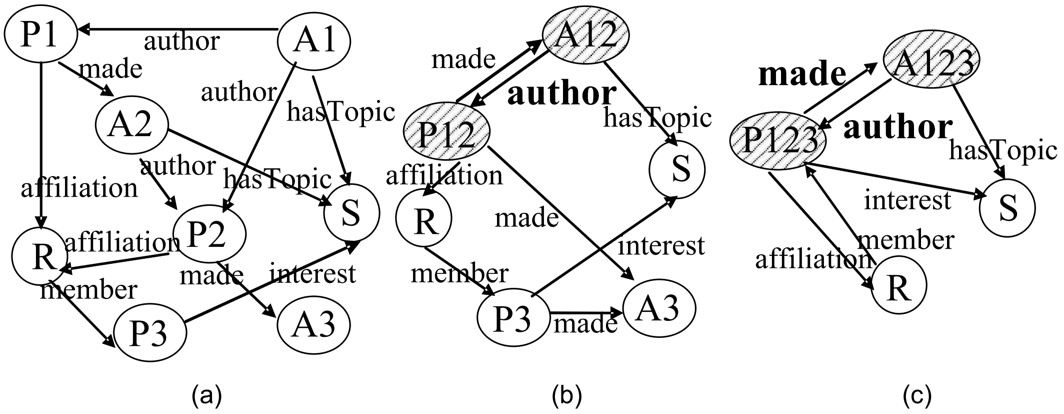


Figure 6.2: An example process building a summary graph. The summarized predicates are highlighted and the summarized nodes are shaded. (a) The original RDF graph. (b) An intermediate state of building the summary graph where the predicate *author* is summarized. (c) An intermediate state of building the summary graph where the predicate *made* is summarized.

of some instantiations of premise is the same as the object of instantiation of the conclusion. When applied to the summary graph, this condition means there is a semantic connection, where is expressed as a path of multiple edges on the summary graph that connects two end points of a single edge on the summary graph (i.e. a relation). Then if we find such a case on the summary graph, there could be a semantic dependency whose premise is that semantic connection and conclusion is the direct relation. For example, the semantic connection consisting of *made* and *hasTopic* on Fig. 6.2 (c) connects the two end points of the *interest*. So a candidate SD could be $\textit{made} \circ \textit{hasTopic} \rightarrow \textit{interest}$. Note the direction on the summary graph only reflects the predicate used in real data, however the semantics of the inverse relation of that predicate is also demonstrated, even if the inverse property is not explicitly defined in data. For example, $\textit{hasTopic} \circ \textit{interest}^{-} \rightarrow \textit{author}$ is also a candidate SD though the inverse of *interest* may not be defined in the original data set. Based on this point, a SD is embedded as a cycle on the undirected summary graph. Then the algorithm finding all possible SDs is transformed to find all undirected cycles on the summary graph and then recover the directionality of properties in the premise according to the direction of conclusion. The algorithm to determine if these candidate SDs found are truly SDs will be given in next subsection.

To detect all cycles in the summary graph, I used Mateti and Deo's [66] algorithm. The algorithm has three main stages. The first is a standard depth first search (DFS) starting with any node which outputs a spanning tree (or more if the graph is not connected). Based on this spanning tree, the edges of the original graph are in two types: forward edges, which point from a node of the tree to one of its descendants and belong to the spanning tree itself, and back edges, which point from a node to one of its ancestors. For example, in Fig. 6.3, given graph (a), DFS will generate a tree (b) and three back edges (6,4), (6,3) and (6,2). Second, given the spanning tree(s), each cycle corresponding to each back edge is found. All these cycles are collected in a cycle base with size m - (n - 1), where m and n are the number of edges and nodes in a graph, because n-1 edges in spanning tree and all others are back edges. The cycles in a cycle base are independent in the sense that no one cycle in the set can be constructed by the union (defined below) of two or more other cycles of the set. The cycle base is not unique, because we can get different trees by starting the depth first search from different nodes. In Fig. 6.3, graph (c), (d), and (e) are cycles corresponding to three back edges found previously and they form into a cycle base. Finally, every other cycle of the graph (any one not in the cycle base) can be obtained by the combination of two or more cycles in the cycle base. The combination operation is an bit XOR operation by representing cycles in edge incidence vectors. A edge incidence vector use each edge on the graph as the element of a vector. In this example, if the edge incidence vector is [(1,2) (2,3) (2,6) (3,4) (3,6) (4,5) (4,6) (5,6)], then the cycle (d) is represented as [00011101] and (e) [01110101]. So the union of them will produce [01101000] which corresponds to cycle (2,3,6,2) (shown as Fig.6.3 (f)). One special case is that an edge is from and to the same node, i.e. self-loop. In that case, I can just add this self-cycle edge any times into the cycles the algorithm found to form a new cycle, if the original cycle contains the node that the special edge is on. Because there is a length limit for a semantic connection, the cycles with self-loop edges are also limited. Although in theory, there is no limit on the length of a semantic connection (Definition 12), its semantic meaning becomes vague if too many relations are joined between a pair

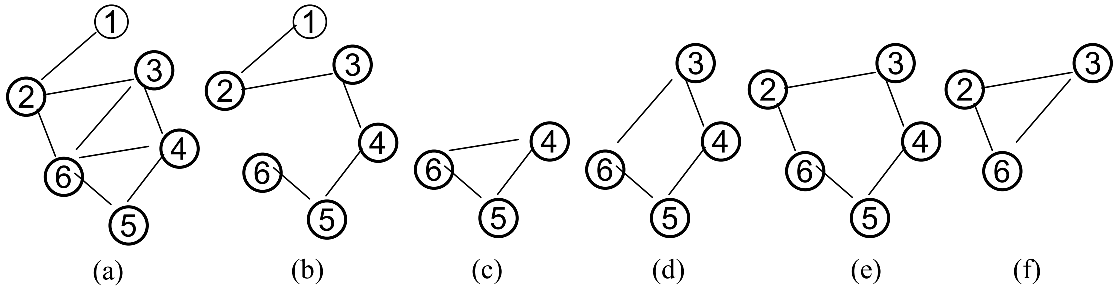


Figure 6.3: An example process of finding all cycles in an undirected graph. (a) is the graph. (b) is the spanning tree created by a DFS starting at node 1. (c), (d) and (e) are three cycles for three back edges found in DFS. (f) is a cycle obtained by combining cycle (d) and (e).

of objects. Since the semantic connections found by the system are used to determine the dependency among relations between objects, it would be unreasonable if no length limit is set. According to my observations and discussions in previous chapters, I set the maximum length of a semantic connection as five in this work as well.

I use summary graph to detect candidate semantic dependencies, but not all of these correspond to real semantic dependencies, some are artifacts of the abstraction process. In other words, a candidate SD is true only if there are instantiations in the data set. Since each SD is a cycle on the summary graph consisting of its premise and conclusion, a SD is a special semantic connection whose head and tail are the same. Thus the function *Inst()* to find instantiations of a semantic connection is generic for a whole SD or its premise only (i.e. a semantic connection). Each semantic connection is computed by the function once and stored for later use. The function $Inst()$ returns all the Paths that are instantiations of the semantic connection. This function can efficiently be computed by using SPARQL queries.

6.3 Probability of Semantic Dependency

Given the output P_E (pairs of triple / prior probability) returned by function *Prior_Prob()*, this section will introduce the function *Get_Prob_SD()* to determine

SD's probability and the function to refine the belief of SDs. The final part of this section is about the function *Get_Prob_Triple()* which computes the posterior probability of triples based on SDs.

6.3.1 Computing Probability of a Semantic Dependency

Each cycle the system found on the summary graph can be interpreted as multiple SDs. For example, in Fig. 6.2 (c), the cycle consisting of *make*, *topic* and *interest* can be interpreted as three SDs whose conclusions are these three relations respectively. Intuitively, the conclusion of *interest* derived from the premise *made* $\circ$ *hasTopic* might be more believable than the conclusion of *made* derived from *interest* $\circ$ *hasTopic*$^{-}$. The analysis under the intuition is that the premise is more specific than the conclusion and there are less counter examples. Thus the probability of SDs in the same cycle should be different and I consider the following in its computation. First, the more the instantiations of a SD there are, the more believable a SD is. Second, each time the premise is instantiated, but the conclusion not, decrease our belief in this SD, because we have found a counter example. Third, the belief of triples involved in these instantiations affects the belief of a SD. Therefore, taking the triples and their prior probabilities as inputs, the probability of a SD s is defined as equation 6.1, which is the sum of the probability of SD instantiations divided by the number of premise instantiations. The function *Inst()* is introduced earlier and *LHS()* is shown in Definition 17 previously. P_E is a vector of probabilities for each triple. The common naive Bayes assumption is used here. Since the product of the probability of triples on a Path is less than or equal to 1 and the number of Paths in *Inst(s)* is less than or equal to those in *Inst(LHS(s))* because the former is a stronger constraint than the latter, the equation guarantees the probability is in [0, 1].

$$Get\_Prob\_SD(s, G, P_E) = \frac{\sum_{i \in Inst(s,G)} \prod_{t \in i} P_E[t]}{|Inst(LHS(s), G)|} \tag{6.1}$$

6.3.2 Refine Probability of a Semantic Dependency

From a logic perspective, the SDs that the system discovered are also probabilistic axioms defined over the original ontology that the data conforms to. In a well-formed ontology, the TBox concepts should be consistent and all concepts and properties should be satisfiable. Thus, based on the explicitly defined original TBox and these probabilistic axioms deduced from the data, it is necessary to do a consistency check on them and accordingly to revise the beliefs on these axioms. The function $Refine\_Prob_SD()$ (line 7 in Algorithm 2 for this purpose will be discussed below.

All SDs can be represented as property chain axioms in description logic (DL) using OWL 2 features[2]. I use the reasoner Pellet[3] to check consistency and satisfiability and use its explanation generator to get the set of axioms causing that problem. An important satisfiability check is about property. For example, if two axioms are $p1 \circ p2 \rightarrow p3$ and $D(p1) \sqcap D(p3) \sqsubseteq \perp$ (i.e. the domains of the property p1 and p3 are disjoint), the unsatisfiability of $p1$ or $p3$ should be detected by such check. However, to the best of my knowledge, no popular DL reasoners support the property satisfiability check. But, an inferred unsatisfiable axiom is an important indication to the necessity of adjusting the belief of the axioms involved in this inference. In addition to using a standard DL reasoner, I include the following customized check for unsatisfiability. I first compute the fixed point over these axioms. Because I only check logic consistency between the domain of the head property (Definition 12) and the conclusion property, between the range of the tail property (Definition 12) and the conclusion property, the computation can be stopped when there is no new head or tail property with respect to any conclusion. Although the disjointness between classes have been shown as an important type of axioms for evaluating ontology consistencies, in real world ontologies they are often missing [93]. So in that case user can customize disjointness axioms in ontologies for this check.

In order to determine the probability of axioms, I treat each as a proposition

[2] http://www.w3.org/TR/owl2-overview/

[3] http://clarkparsia.com/pellet/

that can either be true or untrue in a possible world. I identify each possible world by the set of axioms that are true in it. Having inconsistencies, axioms need to be penalized if they are involved in this inference, which also makes other axioms indirectly rewarded. Every group of inconsistent axioms we tracked is the minimal set, i.e. no proper subset of this group can make this inconsistency. Intuitively, if a possible world that has very small probability, e.g. 10^{-6}, satisfies these axioms, the degree of inconsistency of this group is very low and the axioms in it may not need to be blamed too much. The reason is that in most of possible worlds $(1 - 10^{-6})$, they are satisfiable. Thus, the likelihood of possible worlds, whose set of axioms is inconsistent, reflects the degree of inconsistency of this group of axioms. Therefore I assume that each possible world contribute equally to the degree of inconsistency of all groups of inconsistent axioms that it can entail. For example, if a possible world entails three groups of inconsistent axioms, then each group of axioms get 1/3 of contribution from this possible world. The probability of each possible world is equal to the multiplication of the probabilities of every axiom in it and they naturally sum to 1.

Here is an example of the process of belief revision. There are three axioms, a_1, a_2 and a_3, the probability of them are p_1, p_2 and p_3 respectively. So there are 8 possible worlds in total, from $\emptyset$ with probability $(1\text{-}p_1)(1\text{-}p_2)(1\text{-}p_3)$ to $\{a_1, a_2, a_3\}$ with probability $p_1p_2p_3$. The sum of the possibilities of all possible worlds is naturally equal to one. Suppose we have two inconsistent groups $\{a_1, a_3\}$ and $\{a_2, a_3\}$. The degree of inconsistency of group $\{a_1, a_3\}$ is contributed by two parts. One is the probability of the world $\{a_1, a_3\}$ which is $p_1(1\text{-}p_2)p_3$ and the other is half of the probability of the world $\{a_1, a_2, a_3\}$ which is $p_1p_2p_3$. We only say half of because another inconsistent group $\{a_2, a_3\}$ also can be entailed by this possible world. Thus the degree of inconsistency of $\{a_1, a_3\}$ is $p_1(1\text{-}p_2)p_3 + 1/2\ p_1p_2p_3$.

Since the inconsistency of every group is caused by axioms in it, the degree of inconsistency is partitioned onto each axiom within the group. Therefore, the more groups of unsatisfiable groups that include an axiom, the more the axiom is penalized. The reason is that usually the majority of the world knowledge is compatible, so it is more likely to be erroneous when it conflicts more with other world knowledge. In the above example, since a_3 is involved in more inconsistencies than the other two axioms, it gets more abnormal belief. I subtract these abnormal believes from the probabilities of SDs to adjust the probabilities of SDs.

6.3.3 Triple's Posterior Probability

The purpose of finding semantic dependencies and computing their probabilities is to compute triple scores by checking how these probabilistic axioms support each triple. This is the last step in each iteration of the system. The algorithm is shown in Algorithm 3. For each triple, the system iterates through all the SDs whose conclusion is the same as the predicate of this triple. If the subject and object of this triple appear as the first and last instance of an instantiation of the premise of a SD, this premise instantiation can leads to the conclusion as this triple, which is evidence supporting this triple. The normality score of this triple will be the sum of the probability of all these SDs. Thus the minimum of the score is 0, i.e. no such SDs can support it, and the largest theoretical score is the total probabilities of a set of SDs with a certain conclusion, i.e. all these SDs can support it. Because the scores in this range reflect certain probability of a triple's normality, the larger the score, the more likely the triple is normal. For convenience, we project these values onto the range [0, 1] as probabilities by a normalization with the largest theoretical sum. Finally, the algorithm returns the whole set of triples and each associated with a probability and we put them in next iteration as triples' prior probability.

6.4 Experiments

In the experiments for this algorithm, the data sets I used are the same as I chose for previous algorithms (Section 5.3). But I didn't split the dataset into training and testing subsets, because this system does not require a high quality training data set in advance. Using the process described in Section 5.3, abnormal triples are created and added to the data set. Neither negative or positive triples are labeled. Datatype properties are omitted from the data set because the algorithm

Algorithm 3 $Get\_Prob\_Triples(P_S, G)$, P_S is a set of pairs $\langle s, p \rangle$, i.e. a SD and its probability, $G = (I, L, E, R)$ is an RDF graph.

```
$\forall e \in E, P_E[e] \leftarrow 0$
**for** each $t = \langle sub, pred, obj \rangle \in E$ **do**
    $P = 0$
    **for** each $\langle s, p \rangle \in P_S$ and $RHS(s) = pred$ **do**
        **for** each c $\in Inst(LHS(s), G)$ **do**
            **if** $First(c) = sub$ and $Last(c) = obj$ **then**
                $P = P + p$
    $P_E[t] \leftarrow P$
$P_E = P_E / \max_{e \in E} P_E[e]$
**return** $P_E$
```

Table 6.1: Several example semantic dependencies in SWRC and DBpedia.

Semantic Dependency and its Description
made$^-$ ∘ affiliation ∘ member ∘ maker$^-$ ∘ hasTopic → hasTopic Colleagues have papers with the same topic.
isPartOf ∘ isPartOf$^-$ ∘ hasTopic → hasTopic Papers that are in the same part of a proceeding have the same topic.
author$^-$ ∘ creator ∘ author$^-$ ∘ made$^-$ ∘ heldBy$^-$ → holdsRole Conferences where people publish have roles similar to those held by their colleagues in other conferences.
publisher ∘ country ∘ language → language The language of a publisher's country is the same as the work's language.
parentCompany ∘ keyperson → owner The key person of the parent company is the owner of this company.

mainly focuses on object property triples. After running the system, every triple is output with a score. The results are analyzed mainly from a query perspective, i.e. given a credible threshold how many positive triples have scores that are above it and how many negative triples have scores that are below it. Ideally, the system is expected to differentiate all triples into two disjoint sets, the normal triples and the abnormal triples, where the scores of triples in the normal set are higher than the credible threshold which is higher than the scores of the triples in abnormal set. I use standard information retrieval metrics for evaluation. Further I take my previous system (Chapter 5) as the baseline system to compare. The detailed baseline system

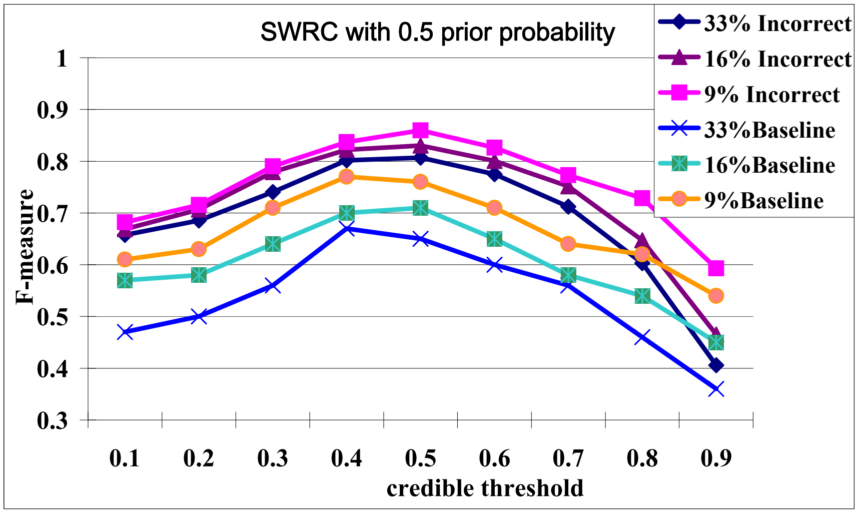


Figure 6.4: The effect of different percentage of abnormal data in SWRC and different credible relation threshold.

configuration will be given when I discuss the results.

6.4.1 Results

Before looking at the performance on differentiating triples, I first show several top ranked interesting SDs in SWRC and DBpedia in Table 6.1. In the SD on the first row, the LHS of this SD says the following. A paper P is made by a person who has a certain affiliation and that affiliation has a member who is the maker of another paper which has a certain topic. The RHS of it says that the paper P also has that topic. Thus the underlying meaning is that colleagues often have papers with the same topic (shown as description in the table). I note that these SDs may not always be true, but most of them do give some sense about the expected context for a triple. However, some SDs also are hard to interpret the meaning. For example $genre^{-} \circ artist \circ musicalArtist^{-} \circ writer \rightarrow composer$. The situation here may be unnecessarily complex SDs. Interesting future work would be to use simple SDs to reduce complex SDs.

First, I show the system's performance on data sets with different percentages of abnormal data. All triples are assigned prior probability 0.5 and the stopping

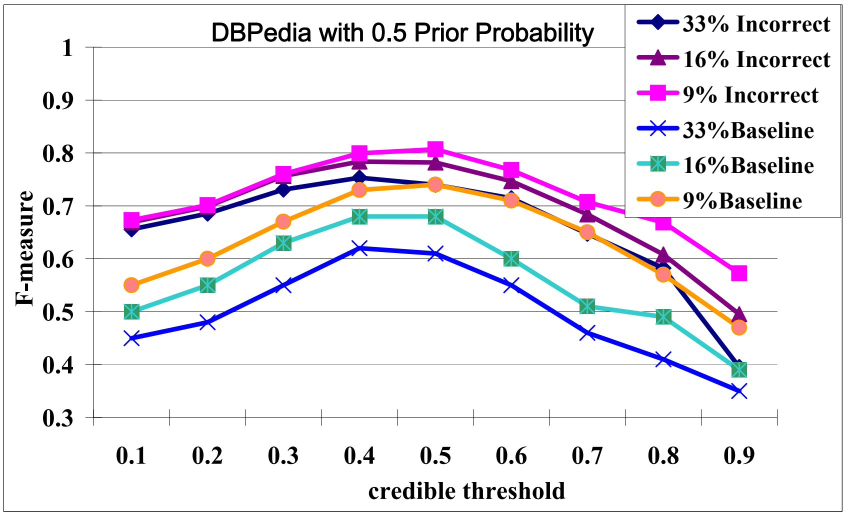


Figure 6.5: The effect of different percentage of abnormal data in DBpedia and different credible relation.

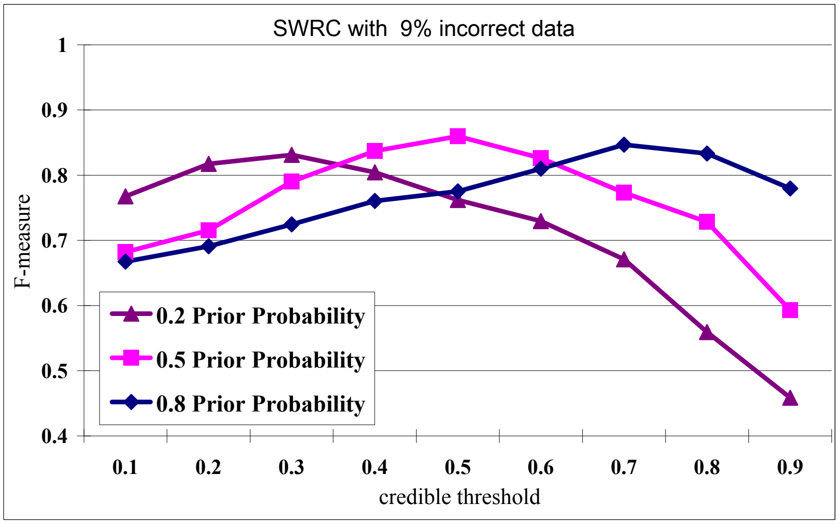


Figure 6.6: The effect of different prior probability assignments in SWRC.

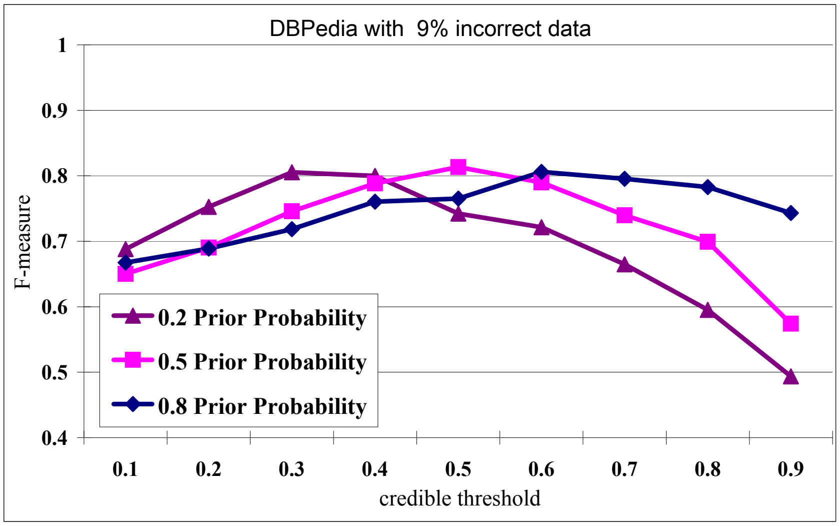


Figure 6.7: The effect of different prior probability assignments in DBpedia.

threshold is set 0.01. We tested three ratios of abnormal triples to normal triples, 1 to 10 (9%), 1 to 5 (16%) and 1 : 2 (33%). From Fig. 6.4 and Fig. 6.5, we see the loss on the best performance in each case is less than the corresponding increase of the number of abnormal triples, e.g. there is only 3% loss on the best F-measures from 16% to 33% abnormal triples on SWRC and 4% on DBpedia. In addition, I compared with my previous system (chapter 5) which is shown as a baseline in the figure. Because the previous system needs training in advance, the baseline is running the previous system by using half of the data set as training and the other half as testing set. I show the baseline system on the data set with 9%, 16% and 33% abnormal data, respectively. Thus comparing between baseline and this system, both running on the data set with 33% incorrect data, it can be observed that my previous system performs at least 10% worse than this system in two data sets. The reason is that the previous system simply assumes that most data in the training data is correct and patterns learned from the training data are all true. Therefore when contexts of incorrect data in the test set exhibit the patterns similar to the training set, the previous system considers them credible and does not label it as incorrect.

Second, I investigated the effect of different prior probability assignments. From

Table 6.2: The effect of different stopping thresholds on the system in SWRC.

α Threshold	0.001	0.005	0.01	0.05	0.1
time (hours)	5.8	3.9	2.9	1.8	0.8
iterations	28	19	13	8	3
F-measure	88.16%	87.44%	85.97%	81.71%	68.65%

Fig. 6.6 and Fig. 6.7, we see the difference on the highest score with different prior probability assignment on data set with 9% incorrect data varies from 82% to 85% on SWRC and from 80% to 81% on DBpedia. Note that for both data sets, higher initial probabilities require higher threshold to maximize the F-measure. The reason is that all triple scores are adjusted based on that initial value and it is natural that the scores of abnormal triples go down due to penalties and others go up due to the support of SDs. This is important in that users of the system can take the credible threshold same as the prior probability that they set as the expected best threshold. In addition, the best scores on two data sets with prior probability 0.5 are a little bit higher than all others. It probably means that this configuration can give better space for the system to adjust the triple probability up or down. But it also can be observed that when the prior probability is 0.8, there is less variation in the F-scores, and on average are better than those for different prior probabilities, since this number is the most close to the actual percentage of correct triples which is 0.91 here.

Recall the system will iteratively refine the probabilities until no triple's probability changes more than a threshold α. The third group of experiments shows the effect of the stopping threshold on the system, such as the number of iterations, the running time and the F-measure. These tests are done on the SWRC data set with 0.5 prior probability. In Table 6.2, we see the F-measure increases when the stopping threshold decreases. The system can achieve a good F-score in reasonable time length and the stopping threshold does not need to be too small.

Finally to test the system in detecting true errors in original data set, I ran the system over the original DBpedia data set without injecting any synthetic incorrect data. The system is set with prior probability 0.5 and stopping threshold 0.005.

Credible Threshold	Number of Triples Reported	Estimated Precision
0.5	53,271	2%
0.1	8,739	13%
0.05	1,635	15%

Table 6.3: The impact of credible threshold on the estimated precision of true errors reported by system.

However, since it is impractical to manually check all of the abnormal data reported by the system, I verified some of these triples (100). Table 6.3 shows the performance of system on reported true errors. It can be seen that when the credible threshold is lower, the precision is higher. However, the precision is very low even for low thresholds. The reason could be that SDs do not have a measure of how broadly each rule can be applied, which is similar to the measure of support used by association rules. Thus it is possible in next algorithm to focus on stronger rules that are applicable more broadly.

To conclude this chapter, I summarize major observations in experiments as follows. The main research problem that this system is trying to solve is to discover useful patterns and detect quality issues without training in advance. The experimental results demonstrated that the system can deal with a data set with a significant portion of incorrect data (e.g. 33%) and accurately capture real patterns underlying the correct data. In addition, the semantic dependency rules can be helpful in many other applications, e.g. enriching or refining the ontology. Through the final experiment on the data set not containing synthetic incorrect data, the system again proves its capability to detect quality issues in Semantic Web data, although with not quite high precision on true errors in real data. However following the similar ideas with previous systems, the rules discovered by this system still rely on the explicit connection (i.e. reusing values among predicates) between the premise and conclusion of this rule. It limits the triples in the evaluation to be object property triples mostly. In addition, the LHS of a rule is a single semantic connection so far. Therefore an immediate research question is whether there is a way to expand these limits so that the rules become more general and the system can detect more errors in not only object property triples but also datatype property triples.

Chapter 7

Detecting Abnormal Data using Value-clustered Graph Functional Dependency

In the previous chapters, I described three systems that can detect erroneous object property triples in different real world scenarios. Meanwhile, I have refined and clarified the forms and semantics of patterns of normal data. The system developed in Chapter 6 succeeded in detecting abnormal data in the data set to be evaluated without any learning. It is also confirmed that some erroneous triples can be found in real world data sets. However, the patterns discovered in the previous system are mainly based on explicit semantic connections (Definition 12) in data which is through reusing values. This point also makes these patterns limited in applicability and only strong enough to report abnormal data by essentially majority voting, but perhaps it would be better to discover higher confidence rules and check them for violations. Thus we need an iterative approach to detect some of abnormal data that may have relatively more explicit anomalous characteristics (e.g. the synthetic incorrect data). To make the approach more efficient (i.e., to avoid iterative process) for detecting more implicit abnormal data, we need to find more, stronger and implicit patterns, compared to the previous systems. The patterns used in my previous work are semantic dependencies (Definition 17). I compared and contrasted semantic dependency rules with association rules in databases. Generally, association rules are dependencies that apply for particular values of some attributes. There is another more common dependency in databases, functional dependency, which is formally defined as follows [24]. Given a relation R, a set of attributes X in R is said to functionally determine another attribute Y, also in R, (written $X \rightarrow Y$), if and only if each X value is associated with precisely one Y value. An example FD $zipCode \rightarrow state$ means, for any tuple, the value of $zipCode$ determines the value of *state*. Functional dependency is devised to specify missing semantics in mere syntactic definitions of database relations [32], and compared to association rules, they are dependencies that are valid for all values of some attributes [5]. Thus, functional dependencies are stronger and never spurious. Therefore, it is possible to detect abnormal data by checking conflict with a few functional dependencies, instead of majority voting using dependencies that are similar to association rules. Then, it would be easier and clearer to explain the reason why the reported data is abnormal. This leads me to consider exploring the concept of finding the equivalent of functional dependencies in RDF graphs.

7.1 Functional Dependency

The rules that my system tries to learn will eventually be used to detect abnormal data that conflicts or does not satisfy them to some extent. From that perspective, these rules are similar to integrity constraints. Functional Dependency is by far the most common integrity constraint for databases in the real world. They are very important when designing or analyzing relational databases. Furthermore, according to my discussion in the beginning of this chapter, the idea of functional dependencies is a potentially fruitful direction for improving the patterns (e.g. semantic dependencies) that are defined and used in my previous work. Thus I will review the concept and techniques related to functional dependencies in this section.

Armstrong [8] has defined useful axioms for inferring functional dependencies. Given that X, Y, and Z are sets of attributes in a relation R, one can derive several properties of functional dependencies. Among the most important are Armstrong's axioms, which are used in database normalization:

- Subset Property (Axiom of Reflexivity): If Y is a subset of X, then $X \rightarrow Y$
- Augmentation (Axiom of Augmentation): If $X \rightarrow Y$, then $XZ \rightarrow YZ$
- Transitivity (Axiom of Transitivity): If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

From these rules, we can derive these secondary rules:

- Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
- Decomposition: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
- Pseudotransitivity: If $X \rightarrow Y$ and $WY \rightarrow Z$, then $WX \rightarrow Z$

Most database researchers agree with these rules. For example, Fagin [31] proves that 1) Armstrong's Dependency Axioms are complete for dependency statements in the usual logical sense that they are strong enough to prove every consequence, and that 2) Armstrong's Axioms are also complete for implicational statements in propositional logic.

Most approaches for finding FD [64, 49, 63] are mainly based on the concept of an agree set [14]. Given a pair of tuples, the agree set is all the attributes for which these tuples have the same values. Since the search for FDs occurs over a given relation and each tuple has at most one value for each attribute, then each tuple can be placed into exactly one cluster where all tuples in the cluster have the same agree set with all other tuples. However, RDF data is not organized in the form of tuples and it is the extensions of RDF properties, which are equivalent to relations with only two attributes (i.e. the subject and object of the triple). To check the value agreement across these relations, each property value may not be placed in one and only one cluster. Thus agree sets are not very useful when applied to RDF data. If all properties are single valued, we could create a tuple and find an agree set for it. However many properties in RDF data are multi-valued and so the correlation between values of different properties becomes more complex. Finally, since most RDF properties are designed just for a subset of instances in the data set, an agree set-based approach will cluster many instances based on null values alone. Finally, all these approaches do not find semantics of property values by further explore property values of them. mapping values of immediate property for a given set of instances, i.e. they do not consider some steps further. For example, they do not consider the semantics of values of property $father$ by further exploring property $brother$ usages on these values. Thus they cannot discover the relationship of $father \circ brother$ (i.e. $uncle$) of original instances. However this characteristic of path of properties is very important for RDF graphs.

RDF graphs are more like graph database models. When both the class schemas and the instances in an object-oriented data model are interpreted as digraphs, the value functional dependency (VFD) [59] defined for the object-oriented data model can have multi-valued properties on the right-hand side, e.g. $title \rightarrow authors$. However the dependencies we envision can have multi-valued properties on both sides and our system can determine the correlation between each value in both sets. Back to 1990s, some researchers realized several problems with the relational model when used for complex applications [54]. Some of these problems derive from its notion of a property, and its strict separation of objects that must have property values, called tuples, from objects that can be property values, called domain values. An important consequence is that query languages which are variations of a "typed" form of the tuple calculus, such as SQL or QUEL, require all terms to denote objects that are domain values. This implies that users must introduce properties of objects to serve as their means of reference, and that all relationships between objects must be expressed indirectly in terms of these properties. Therefore some researchers tried to combine the separate notions of domain and relation into a single notion of class, and thereby allowed properties to be defined between any pair of classes. Terms in query languages are then permitted to traverse any number of properties: none, in recognition that objects that were tuples have separate identity, or more than one, since objects that were domain values are now permitted to also have structure. This data model is called the semantic data model. The path functional dependency (PFD) [98] defined for semantic data models considered multiple attributes on a path, however the PFD did not consider multi-valued attributes. FD$_{XML}$ is the FD's counterpart in XML [58] where its left-hand side is a path starting from the XML document root which essentially is another form of a record in a database. Hartmann et al. [45] proposed a framework allows users to define functional dependencies similar to those in previous works, but enables users to capture further kinds of functional dependencies which happen to be useful in designing XML documents.

The basic equality comparison of values used in FD is limited in many situations, for the following three reasons. Consider (1) for floating point numbers, rounding and measurement errors must be considered. For example, there might be no much difference in semantics between 1.000001 and 1.0. (2) Sometimes dependencies are probabilistic in nature, and one-to-one value correspondences are inappropriate. For example, the days to process an order for shipping a given product is usually limited to a small range as opposed to an exact value. (3) Sometimes certain values can be grouped to form a more abstract value. For example, the literal values "dark blue", "light blue" and "sky blue" might can be considered as a group of values with semantics of "blue". Instead of the basic equality relation between values, a simple extension of relation between values could be by using algebraic operation. Algebraic constraints [21, 43] in database relations are about the algebraic relation between two columns in a database and are often used for query optimization. The algebraic relation can be $+, -, \times, /$. However these algebraic constraints are limited to numerical attribute values and the mapping function can only be defined using one of the four algebraic operators. As the example given above, on the one hand, the number of days to process an order for shipping is not an direct algebraic function relationship with the given products. On the other hand, the product is not a numeric property. Thus algebraic operations between two columns are limited. The reason is that numerical columns are more often indexed and queried over as selective conditions in databases than strings. In contrast, I try to find a general mapping function between the values of different properties, both numbers and strings. Additionally, for the purpose of query optimization, they focus on pairs of columns with top ranked relational significance (based on either by workload profiling or query feedback), i.e. the major parts in each of these pairs and the data related to dependencies that is often queried over, rather than all possible pairs of properties and all pairs of values existing in the data set.

FD can be expressed as a special form of multivalued dependency (MVD). The formal definition of MVD is given below (chapter 7 of [87]). The multivalued dependency $\alpha \twoheadrightarrow \beta$ (which can be read as α multidetermines β) holds on R if, in any legal relational table r that consists of attributes from R, for all pairs of tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$ (the projection of two tuples on attributes α), there exists tuples t_3 and t_4 in r such that

1. $t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$
2. $t_3[\beta] = t_1[\beta]$
3. $t_3[R - \beta] = t_2[R - \beta]$
4. $t_4[\beta] = t_2[\beta]$
5. $t_4[R - \beta] = t_1[R - \beta]$

I give a brief explanation of this definition. The precondition is that two tuples t_1 and t_2 agree on the set α of attributes from R (i.e. both have the same values of these attributes) where α is the set of attributes of the LHS of this multivalued dependency. Then according to the semantics of multivalued dependency, there exists another pair of tuples t_3 and t_4 which satisfy the five conditions above at the same time. The first condition is that all these four tuples agree on the set α of attributes. Together the second and the third condition means tuple t_3 has the same values as t_1 on attributes β while it also has values same as t_2 on all the other attributes. In other words, the tuple t_3 can be seen as built by copying all values from the tuple t_2 and then replacing values of attributes β with corresponding values from the tuple t_1. Similarly, combining condition 4 and 5, the tuple t_4 can be seen as built by copying all values from the tuple t_1 and then replacing values of attributes β with corresponding values from the tuple t_2. Overall, the semantics of these conditions is that if whenever we have two tuples (t_1 and t_2) of R that agree in all the attributes of α, then we can swap their β components and get two new tuples (t_3 and t_4) that are also in R. For example, relational table *Students* has attributes *name*, *address*, *phones*, *school* and a MVD $name \twoheadrightarrow phones$. If *Students* has the two tuples:

```
   name   |  addr  |  phones  |  school  |
============================================
   tom    |   xyz  |    123   |   abc    |
   tom    |   xyz  |    456   |   def    |
```

it must also have the same tuples with values of attribute *phones* swapped:

```
   name   |  addr  |  phones  |  school  |
============================================
   tom    |   xyz  |    123   |   def    |
   tom    |   xyz  |    456   |   abc    |
```

In a functional dependency, each value on the LHS is precisely associated with a single value on the RHS. Compared to FD, in a multivalued dependency, each value on the LHS is precisely associated with multiple values. Note it does not mean that each value on the LHS can be associated with only some one of values from the set of multi-determined values on the RHS. Instead, it means that each value should co-occur with every one multi-determined values in the relation. The more precise the dependency is, the easier detecting abnormal data is. Therefore MVD is useful in database design. But in this work the main goal of discovering dependencies is to find abnormal values that are not expected. Given multiple expected values through using MVD, it would be harder to determine which one is actual abnormal. Further, if the concept similar to MVD is considered in this work, the complexity of system will be greatly increased, especially considering the large scale of Semantic Web data.

Data dependencies have recently shown promise for data quality management in databases. Bohannon et al. [19] focuses on repairing inconsistencies based on standard FDs and inclusion dependencies (INDs), that is, to edit the instance via minimal value modification such that the updated instance satisfies the constraints. They proposed a repair framework that deals with both FDs and INDs. The cost of an attribute-level modification in a repair is essentially the weight of the changed tuple times the distance according to a similarity metric between the original value of the attribute and its value in the repaired database.

A conditional functional dependency (CFD) [34, 25] is more expressive than a FD because it can describe a dependency that only holds for a subset of the tuples in a relation, i.e., those that satisfy some condition. Fan et al. [34] gave a theoretical analysis and algorithms for computing implications and minimal cover of CFDs. In contrast to traditional FDs that were developed mainly for schema design, CFDs aim at capturing the consistency of data by enforcing bindings of semantically related values. For static analysis of CFDs we investigate the consistency problem, which is to determine whether or not there exists a nonempty database satisfying a given set of CFDs, and the implication problem, which is to decide whether or not a set of CFDs entails another CFD. They showed that while any set of transitional FDs is trivially consistent, the consistency problem is NP-complete for CFDs. For the implication analysis of CFDs, they provided an inference system analogous to Armstrong's axioms for FDs, and showed that the implication problem is coNP-complete for CFDs in contrast to the linear-time complexity for their traditional counterpart. The CFD discovery problem has high complexity; it is known to be more complex than the implication problem, which is already coNP-complete [34].

Cong et al. [25], similar to Bohannon et al., focused on repairing data on two central criteria for data quality: consistency and accuracy. Inconsistencies and errors in a database often emerge as violations of integrity constraints. Given a dirty database D, one needs automated methods to make it consistent, i.e., find a repair D' that satisfies the constraints and "minimally" differs from D. Equally important is to ensure that the automatically-generated repair D0 is accurate, or makes sense, i.e., D' differs from the "correct" data within a predefined bound. This paper studies effective methods for improving both data consistency and accuracy. Cong et al. employed a class of CFDs to specify the consistency of the data, which are able to capture inconsistencies and errors beyond what their traditional counterparts can catch. To improve the consistency of the data, they proposed two algorithms: one for automatically computing a repair D0 that satisfies a given set of CFDs, and the other for incrementally finding a repair in response to updates to a clean database. However shown in the work, both problems are intractable.

In contrast to the above works using FDs, INDs or CFDs to repair databases, we are trying to both automatically find fuzzy constraints/dependencies, i.e. those that hold for most of the data, and report on exceptional data for applications. Our work incorporates advantages from both FD and CFD, i.e. fast execution and the ability to tolerate exceptions.

7.2 Value-clustered Graph Functional Dependency

RDF data also has various dependencies. But RDF data has a very different organization and FD cannot be directly applied because RDF data is not organized into relations with a fixed set of attributes. We propose *value-clustered graph functional dependency* (VGFD) based on the following thoughts. First, FD is formally defined over one entire relation. However RDF data can be seen as extremely decomposed tables where each table is a set of triples for a single property. Thus we must look for dependencies that cross these extremely decomposed tables and extend the concept of dependency from a single database relation to a whole data set. Second, the correlation between values is trivially determined in a database of relational tuples. But in RDF data, it is non-trivial to determine the correlation, especially for multi-valued properties. For example, in DBpedia, the properties *city* and *province* do not have cardinality restrictions, and thus instances can have multiple values for each property. This makes sense, considering that some organizations can have multiple places. Yet finding the correlation between the different values of *city* and *province* becomes non-trivial. Third, traditionally value equality is used to determine FD. However, as discussed when introducing functional dependencies, this is not appropriate for real world, distributed data.

Following the thoughts above, I give the formal supporting definitions of VGFD as follows.

Definition 18. *Given an RDF graph $G = (I, L, R, E)$, a Composite Property r° in graph G is $r_1 \circ r_2 ... r_n$, where $r_i \in R$ or $r_i^- \in R$, and $\exists I_0, ..., I_n$, $\langle I_0, r_1, I_1, ..., r_n, I_n \rangle$ is a Path in G. Let $\mathcal{R}^\circ$ be all possible Composite Properties. Given $r^\circ \in \mathcal{R}^\circ$, $Triple(r^\circ, G) = \{\langle I_0, r^\circ, I_n \rangle | \langle I_0, r_1, I_1, r_2, I_2, ..., r_n, I_n \rangle \textit{ is a Path in } G\}$. $Length(r^\circ) = n$. $\forall r \in R$, $r \in \mathcal{R}^\circ$ and $Length(r) = 1$.*

Definition 19. *Given an RDF graph $G = (I, L, R, E)$, a Conjunctive Property r^+ in graph G is a set $\{r_1, r_2, ..., r_n\}$ (written $r_1 + r_2 + ... + r_n$), where $\forall r_i \in r^+, r_i \in \mathcal{R}^\circ$ and $\exists I'$, s.t. $\forall 1 \leq i \leq n$, $I_i \in I$, $\langle I', r_i, I_i \rangle \in Triple(r_i, G)$. Let $\mathcal{R}^+$ be all possible Conjunctive Properties. $Size(r^+) = \sum_{r_i \in r^+} Length(r_i)$.*

A Composite Property is a sequence of edges on a Path. The subject and object of a Composite Property are the first and last objects on the Paths consisting of this sequence of edges. Every original property in the data is a special case of Composite Property whose length is one. A Conjunctive Property groups a set of Composite Properties that have a common subject I'. Every Composite Property can be seen as a special form of a Conjunctive Property which has a single Composite Property in the set. Thus, each original $r \in R$ is also $r \in \mathcal{R}^\circ$ and each $r^\circ \in \mathcal{R}^\circ$ is also $r^\circ \in \mathcal{R}^+$.

Definition 20. *Given an RDF graph $G = (I, L, R, E)$, $i \in I$, and $r^\circ \in \mathcal{R}^\circ$, the value function V° is defined as $V^\circ(i, r^\circ) = \{i' | \exists \langle i, r^\circ, i' \rangle \in Triple(r^\circ, G)\}$.*

Definition 21. *Given an RDF graph $G = (I, L, R, E)$, $i \in I$, and $r^+ \in \mathcal{R}^+$, the value function V^+ is defined as $V^+(i, r^+)$ is a tuple $< V^\circ(i, r_1), ..., V^\circ(i, r_1) >$ where $\forall j, r_j \in \mathcal{R}^+$.*

Given a Composite Property and a given subject, value function V° returns the set of objects connected with this subject through this Composite Property. Given a Conjunctive Property and a given subject, the value function V^+ returns a tuple of sets of objects connected with this subject through this Conjunctive Property.

To illustrate these definitions, Fig. 7.1 gives an RDF graph showing the examples of these definitions. An example Path (Definition 2) is $\langle DarwicheP97, \textit{has-date}, \text{1998-04-03}, \textit{has-year}, 1998\rangle$. Because of this Path, an example of Composite Property can be defined as $\textit{has-date} \circ \textit{has-year}$ and its length is two. Then an example of Conjunctive Property can be $\textit{article-of-journal} + \textit{has-volume}$, because it is composed of two composite properties of length one: $\textit{article-of-journal}$ and $\textit{has-volume}$. Given Composite Property $\textit{has-date} \circ \textit{has-year}$ and instance $DarwicheP97$, the set of values returned by value function of $V^\circ(DarwicheP97, \textit{has-date} \circ \textit{has-year})$ is set consisting of single value 1998. Given Conjunctive Property $\textit{article-of-journal} + \textit{has-volume}$ and instance $DarwicheP97$, the tuple of values returned by value function of $V^\circ(DarwicheP97, \textit{article-of-journal} + \textit{has-volume}$ is $< \{\textit{journal-}297\}, \{6\} >$.

Definition 22. *Given an RDF graph $G = (I, L, R, E)$, $r^\circ \in \mathcal{R}^\circ$ and $x \in I \cup L$, the cluster function is defined as*

$$C(x, r^\circ) = \begin{cases} \textit{the equivalence class of } x, & \textit{when } x \in I \\ \textit{the cluster of values of property } r^\circ \textit{ containing } x, & \textit{when } x \in L \end{cases}$$

Definition 23. *Given $i, j \in I$ and $r^\circ \in \mathcal{R}^\circ$, Dependency Equality (DE) between i and j on r° is: $V^\circ(i, r^\circ) \doteq V^\circ(j, r^\circ) \iff (\forall x \in V^\circ(i, r^\circ) \iff \exists y \in V^\circ(j, r^\circ), C(x) = C(y))$. Given $i, j \in I$ and $r^+ \in \mathcal{R}^+$, Dependency Equality between i and j on r^+ is: $V^+(i, r^+) \doteq V^+(j, r^+) \iff \forall r_k \in r^+, V^\circ(i, r_k) \doteq V^\circ(j, r_k)$.*

Definition 24. *A value-clustered graph functional dependency (VGFD) s in graph G is $X \to Y$, where $X \in \mathcal{R}^+$, $Y \in \mathcal{R}^\circ$ and $\forall i, j \in I$, if $V^+(i, X) \doteq V^+(j, X)$ then $V^\circ(i, Y) \doteq V^\circ(j, Y)$.*

These last definitions complete our definitions of VGFDs. Definition 22 defines the cluster function which returns the cluster that groups a set of property values with close semantics. As I discussed in section 7.1, when considering traditional functional dependency, most approaches uses basic equality to compare object identifier or actual values. In that mechanism, some similar values are considered different so that a meaningful dependency about these values is missed. Therefore in this work,

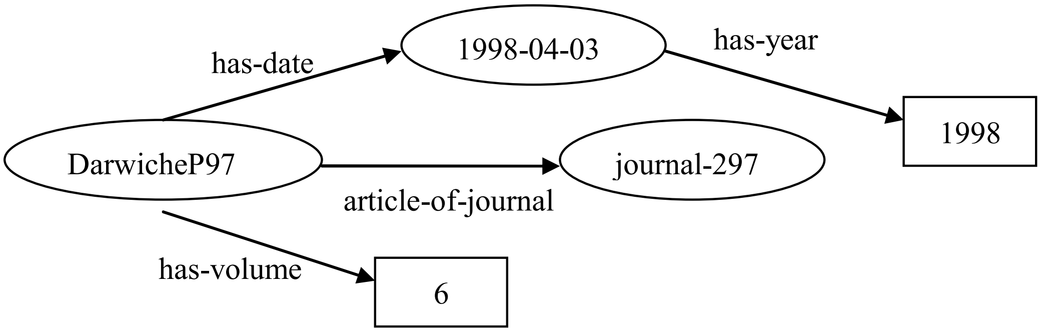


Figure 7.1: An RDF graph example illustrating the definitions related to VGFD.

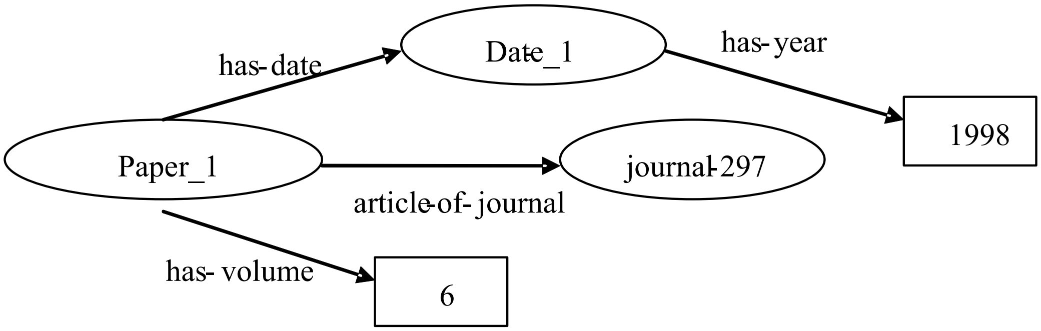


Figure 7.2: Another RDF graph example illustrating the patterns to be discovered by VGFDs.

I considered typical situations where values can be similar enough to still generate a dependency. Specifically, the cluster function 1) returns the sameAs transitive closure for an instance which is involved in some sameAs triples, 2) returns one of the clusters for datatype values of given property and this cluster contains given datatype value, otherwise 3) returns the value itself. The transitive closure of relation R is as follows, intuitively constructing it step by step. To start, define $R^0 = R$ and, for $i > 0$, $R^i = R^{i-1} \cup \{(s_1, s_3) | \exists s_2 \text{ where } (s_1, s_2) \in R^{i-1} \text{ and } (s_2, s_3) \in R^{i-1}\}$. Be specific to our problem, for example, a transitive closure for two sameAs triples $< a\ sameAs\ b >$ and $< b\ sameAs\ c >$ is adding a new triple $< a\ sameAs\ c >$ into these set of triples. Meanwhile, the instances a, b and c forms a equivalence class. Note, when an instance is not involved in any sameAs relation, based on the above notation $R^0 = R$, this instance itself is a equivalence class. Using this cluster function, Definition 23 defines Dependency Equality (DE) among the values of a Conjunctive Property or Composite Property. When different values of a property satisfy Dependency Equality, it means that these values are treated as the same (in semantics) for considering VGFD. Definition 24 states the pre-condition of a VGFD that given any instance, if object values of a given Conjunctive Property for this instance satisfy Dependency Equality, then there is a DE among values of a Composite Property for this instance. Then there is a candidate VGFD whose left-hand side (LHS) is this Composite Property and right-hand side is this Conjunctive Property. I will take Fig. 7.1 and Fig. 7.2 together as a example. In this example, all instances (two instances $DarwicheP97$ and $Paper_1$) have the same values (i.e. satisfying DE) of Conjunctive Property $article\text{-}of\text{-}journal + has\text{-}volume$, and they also have the same values (i.e. satisfying DE) of Composite Property $has\text{-}date \circ has\text{-}year$. Then there is a candidate VGFD whose LHS is this Conjunctive Property and RHS is this Composite Property: $article\text{-}of\text{-}journal + has\text{-}volume \rightarrow has\text{-}date \circ has - year$.

Due to the union rule of Armstrong's axioms (discussed when introducing functional dependencies) used to infer all the functional dependencies, if $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ hold, then $\alpha \rightarrow \beta\gamma$ holds. Therefore, it is enough to define the VGFD whose right-hand side (RHS) is each single element of a set of Composite Properties, instead of a Conjunctive Property, i.e. the whole set of Composite Properties.

7.3 System Overview

Following the definition of VGFD, I developed an algorithm to discover VGFDs [101]. Because the VGFDs to be discovered are used to detect errors in original data, the dependencies that have single named ontology properties as RHS are more important. Currently this work only detects VGFDs whose RHS Composite Property is with length one, i.e. all named properties. Fig. 7.3 and Algorithm 5 shows the work flow of the system.

In the approach, there are two main parts: the VGFD search (Algorithm 4 and will be introduced in the next section) and value clustering (Algorithm 5 will be

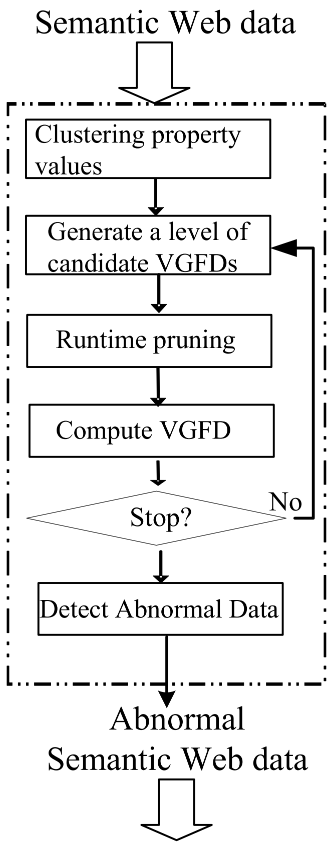

Semantic Web data
Clustering property values
Generate a level of candidate VGFDs
Runtime pruning
Compute VGFD
Stop?
No
Detect Abnormal Data
Abnormal Semantic Web data

Figure 7.3: The work flow of the system based on VGFD.

introduced in the section after) which is used to group values in similar semantics for considering VGFDs. Before the process of discovering VGFDs, the system first cluster property values into sets with similar semantics for considering dependencies on these values.

The main process of this clustering is as follows. The system initializes a set C which is for clusters of property values (line 1). There are several steps to cluster values of each property. First it gets all values for a given property (line 3). The pre-clustering step groups values into coarse grained sets using light weight computation (line 4 to be discussed in section 7.5.1). Based on these pre-clustered groups, the system tries to group values into finer-grained sets and records these resulting sets (line 5 and 6 described in section 7.5.2).

Algorithm 4 $Cluster\_Property\_Values(G, \gamma)$, $G = (I, L, R, E)$ is a graph; γ is the threshold for pre-clustering.

```
$C \leftarrow \emptyset$
for each $r \in R$ do
    $V \leftarrow \{o | \exists < s, p, o > \in E\}$
    $groups \leftarrow Preclustering(V, \gamma)$
    $C_r \leftarrow Optimal\_Kmeans(V, groups)$
    $C \leftarrow C \cup < r, C_r >$
return $C$
```

Algorithm 5 $Search\_VGFDs(G, C, \alpha, \beta)$, $G = (I, L, R, E)$ is a graph; C is set of property/clusters pairs, α is the confidence threshold for a VGFD; β is the sampling size.

```
$S \leftarrow \emptyset$
$i = 0$
$L_i \leftarrow \emptyset$
repeat
    $i = i + 1$
    $L_i \leftarrow Generate\_Level\_with\_Static\_Pruning(L_{i-1}, E)$
    for each $s \in L_i$ do
        if $Runtime\_Pruning(s, \alpha, \beta, E, C) = FALSE$ then
            if $(M \leftarrow Compute\_VGFD(s, \alpha, E, C)) \neq \emptyset$ then
                $S \leftarrow S \cup < s, M >$          //M is the set of value mappings of s.
until $L_i = \emptyset$ or $i >= DEPTH\_LIMIT$
return $S$
```

Given n named properties in data set, there can be $(2n)^k$ composite properties of length k (considering inverse properties). Then there can be $2^{(2n)^k}$ Conjunctive Properties, since we can pick any subset of Composite Properties to compose a Conjunctive Property. Therefore, in the worst case, the number of VGFDs, which has a Conjunctive Property on LHS and a Composite Property on RHS, is $O(2^{(2n)^k} \cdot (2n)^k)$, i.e., super-exponential. In database theory, the closure of a set F of functional dependencies is the set of all functional dependencies logically implied by F. When the closures of two sets of functional dependencies are the same, these two sets of functional dependencies are equivalent and each is called the cover of another other. Therefore to efficiently discover a minimum set of VGFDs which is a cover of the whole set of VGFDs, my approach essentially is computed level-wise. As an example shown in Fig. 7.4, each level L_i consists of VGFDs with LHS of size i (Definition 19). The computation of VGFDs with smaller sets of LHS properties can be used when computing children VGFDs that have a superset of those LHS properties. A similar level-wise search was proposed for the Tane algorithm [49] to discover FDs in a relational table in database. Each node on a certain level on the containment lattice in Tane specified a group of attributes for considering a FD. The group of attributes is only considered as a set without ordering and ways of combination. Thus a set of attributes will be a unique node on that lattice. However the same set of properties can be grouped into different nodes in my level searching process, because there are multiple ways of combine them using the composite and conjunctive operators. Thus, the nodes in my level searching process are finer grained which leads to more opportunities for pruning. Algorithm 5 initializes a set S for recording VGFDs (associated with their value mappings) (line 1), then starts with level 0 (line 2 and 3). On each new level (line 5), it first generates possible VGFDs on this level based on the results of previous levels and it also eliminates many potential VGFDs from further consideration based on some easily computed heuristics (line 6, discussed in Section 7.4.1). Then, for each new generated candidate VGFD on this level, runtime pruning (line 8, discussed in Section 7.4.3) is conducted in order to avoid expensive computation for false VGFDs. If the candidate VGFD has not been eliminated by the previous steps, a detailed computation (line 9, discussed in Section 7.4.2) is conducted. This computation produces the value mappings between the LHS property and RHS property of a VGFD and the pair of the VGFD and its value mappings is recorded for return (line 10). The mappings can be used to detect erroneous triples which do not have a mapped value. The whole process can terminate at a predetermined level, or after all levels, although the latter is usually unnecessary and unrealistic (line 11).

Based on clusters and VGFDs, abnormal data can be in two types: one is far away from other clusters and the other is a violation of VGFDs. Specifically, in this work, a triple is reported as an outlier if its value is the only element of some cluster whose distance to the nearest cluster centroid is above twice of the average distances between all clusters for this property. A triple is reported as abnormal due to violation of VGFDs only when its value conflicts with a value mapping determined by some VGFD and this value mapping is confirmed by other triple usages more than twice.

7.4 Discovering VGFDs

A naive process of discovering VGFDs is to combine all properties in all possible ways on both LHS and RHS of a candidate VGFD and then check if all values of LHS property can functionally determine values of RHS property. That would be too inefficient and actually impractical. Therefore it is necessary to prune out those unlikely combination of properties with minimum computation. I devised a static pruning to rule out some candidate VGFDs with static information (section 7.4.1) and a runtime pruning process to further rule out some candidate VGFDs by checking sample data of this VGFD (section 7.4.3). When actual computing a VGFD, I will discuss the most important aspect in it that is to handle multi-valued properties (section 7.4.2).

7.4.1 Heuristics for Static Pruning

I first define the discriminability for a property as the number of distinct object values divided by the size of the property extension. Then, the static pruning heuristics used to eliminate potential VGFDs from further consideration are:

1. insufficient subject overlap between the LHS property and the RHS property,
2. the LHS property or RHS property has too high a discriminability,
3. the discriminability of the LHS property is less than that of the RHS property.

The information for rule 1 can be acquired from an ontology (e.g. using domain and range information) or a simple relational join on data. Here insufficient overlap means too few common subjects, e.g. less than 20. Because VGFDs essentially are based on patterns during co-occurrences of properties for some instances, few co-occurrences cannot reveal enough information to discover patterns. Co-occurrence is another way of saying of subject overlap among properties.

As defined above, the maximum value of a discriminability is one only when every value of this property usage is different. In rule 2, if the discriminability of a property is close to one, e.g. 0.95 which means that in 95% property usages, the values are different, then the property functions like a superkey in a database, i.e. each record has a unique value for the superkey. Since such keys can identify an individual, they are not very useful for detecting abnormal data by checking its value patterns in usages.

As defined, a VGFD is an extension of FD in RDF graph and so it still follows the idea that values of the LHS precisely determine values of RHS. In other words, there is a functional mapping between LHS values and RHS values. In rule 3, if there is a mapping between two properties where the discriminability of the LHS property is less than that of the RHS property, then some values of the property with smaller discriminability must be mapped to different values on the RHS, which would not be a functional mapping. In order to apply these heuristics, we define the additional observations:

1. The discriminability of a Composite Property is never no greater than that of each property involved.

2. The discriminability of a Conjunctive Property is never no less than that of each property involved.

3. A Conjunctive Property cannot be based on two properties that have few common subjects.

4. A Composite Property cannot be based on two properties that have few common objects and subjects.

5. All children of a true VGFD on the level-wise search graph are also true VGFDs, but are not minimal.

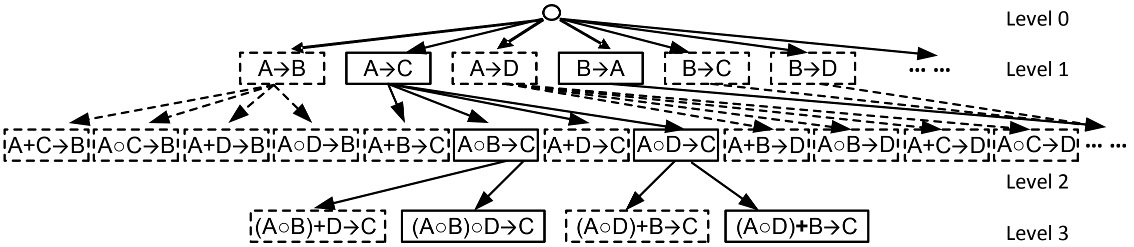


Figure 7.4: An example of level-wise discovering process. We suppose that (1) property A and B have few common subjects, (2) the discriminability of B is less than that of C and (3) D has a high discriminability.

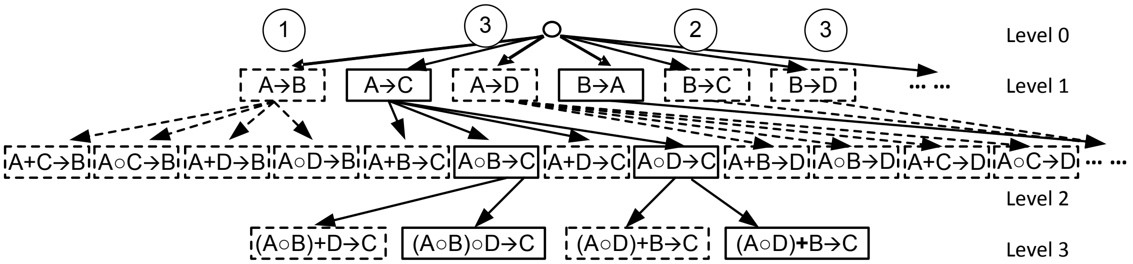


Figure 7.5: Some of the candidate VGFDs on the first level are pruned out.

For example, given a Composite Property $A \circ B$, its values all come from the values of B and its extension is a subset of the Cartesian product between objects of A and subjects of B, then its discriminability, i.e. the distinct values divided by the usages, should be no greater than that of either component. A similar explanation applies for Conjunctive Properties in observation 2. An extension of the observation 4 is that a Conjunctive Property cannot be followed by another property to compose a Composite Property. Take $(A + B) \circ C$ as an example property. Since the values of property $A + B$ are tuples as opposed to the normal instances in RDF data that can be the subjects of other properties (e.g. C).

Fig.7.4 is an example showing how these heuristics are useful in the level-wise searching. Each edge is from a parent VGFD to a child VGFD and the LHS of child VGFD is a superset of the LHS of parent VGFD. Two connected VGFDs have the same RHS. The VGFDs pruned by the above heuristics are in dotted boxes and dotted lines connect parent VGFD and those child VGFDs that are pruned due to the same heuristics as the parent VGFD. For this example, I make assumptions

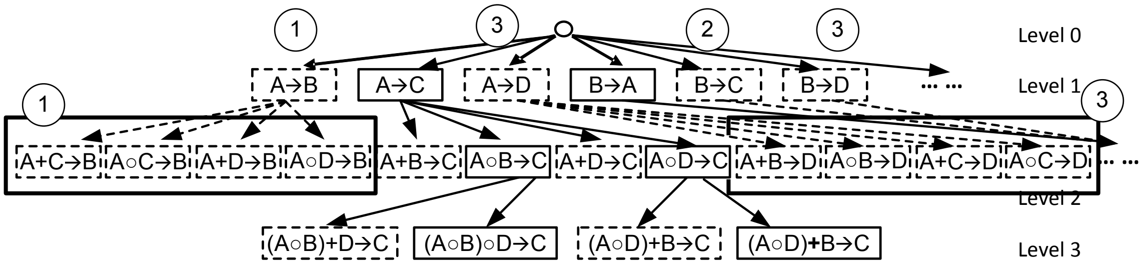


Figure 7.6: Some of the candidate VGFDs on the second level are pruned out due to the reason as their parents.

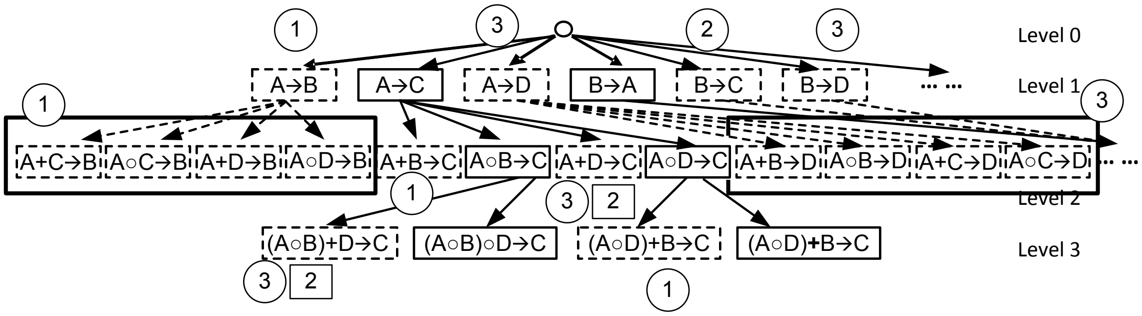


Figure 7.7: Other candidate VGFDs are pruned out.

typical of real RDF data. For instance, in DBpedia less than 2% of all possible pairs of properties share sufficient common instances. So following our heuristics, four VGFDs on level 1 are pruned (see Figure 7.5, the numbers in circle are defined heuristics above): $A \rightarrow B$ is due to heuristic rule 1, $B \rightarrow C$ is due to rule 3 and the other two are due to rule 2. Then the children of $A \rightarrow B$ and $A \rightarrow D$ are pruned due to the same reason as their parents (see Figure 7.6). Shown in Figure 7.7 (where number in box is the defined observation above), $A + B \rightarrow C$ on level 2 and $(A \circ D) + B \rightarrow C$ on level 3 are pruned due to the first assumption plus the observation 2. Finally, $A + D \rightarrow C$ on level 2 and $(A \circ B) + D \rightarrow C$ on level 3 are pruned due to the observation 1 and heuristic rule 2. From this example, we can see simple conditions can reduce the level-wise search space greatly based on these heuristics.

Table 7.1: The left table is the triple list. The right table is mapping count.

deptNo		deptName	
subject	object	subject	object
A	1	A	CS
A	2	A	EE
B	1	B	EE
C	2	C	CS
D	2	D	EE

Candidate Value Mapping	Count
1→ EE	2
2→ EE	2
2→ CS	2
1→ CS	1

7.4.2 Computing VGFDs

If two properties are considered as the LHS and the RHS of a VGFD, it is required to investigate whether all these value pairs show correlations. In the case of both properties are single valued properties, we can create a table and put the value pairs into tuples and simply scan the table to see if every LHS value is only associated with single RHS value. However the fundamental difference between VGFD and FD when computing VGFD is that we consider multi-valued properties. When finding FDs in databases, the multi-valued attributes either are not considered (if they are not in the same relation), or the correlation of their values is given by having separate tuples for each value. RDF frequently has multi-valued properties without any explicit correlation of values, e.g. in DBpedia, more than 60% properties are multi-valued. Therefore I devised the following general approach dealing with both single property values and multi-valued properties. When computing a VGFD, we try to find a functional mapping from each LHS value to an RHS value such that this mapping maximizes the number of correlations. We consider any two values for a pair of multi-valued properties that share a subject to be a possible mapping. Then we greedily select the LHS value that has the most such mappings and remove all other possible mappings for this LHS value. If multiple RHS values are tied for the largest number of mappings, then we pick the one that appears in the fewest mappings so far. Consider Table 7.1 which analyzes the dependency *deptNo* → *deptName*. The triples are given to the left and each possible value mapping and its maximal possible count are listed in descending order to the right. The maximal count of 1 → EE is 2, because these two values co-occur for instances A and B once for each. We first greedily choose mapping 1 → EE, because it has the largest count among all mappings for $depNo = 1$. After this selection, the mapping 1 → CS is removed since $deptNo = 1$ has been mapped. Then for $deptNo = 2$, to maximize the number of distinct values being matched on both sides, we choose $(2, CS)$ since CS has been mapped to by fewer LHS values than EE. The confidence in a VGFD depends on how often the data agree with it, i.e., the total matches divided by the sum of the LHS's extension, e.g. the VGFD above has the confidence of $4/5 = 0.8$. In this work, we set the confidence threshold $\alpha = 0.9$ to ensure that patterns are significant, while allowing for some variation due to noise, input errors, and exceptions.

Note the basic equality used here is a special case of cluster-based Dependency Equality. In the example, we assumed EE and CS were in different clusters. For example, if CS and EE are clustered together into a group of similar semantics (noted as $EECS$), then the mappings will be 1 → $EECS$ and 2 → $EECS$.

7.4.3 Run-time Pruning

As discussed earlier, since in the worst case, there are $2^{(2n)^k} \cdot (2n)^k$ possible VGFDs, the expensive full scan of value pairs must occur many times. So we propose to use *mutual information* (MI) computed over sampled value pairs for estimating the degree of dependency. In Algorithm 6, given a candidate VGFD $s\ X \rightarrow Y$, the system starts with randomly selecting a specified percentage β of the instances. In line 2, for each instance i, the system randomly picks a pair of values from $V^{+}(i, X)$ and $V^{\circ}(i, Y)$. *Distribution*() also applies the clusters C_X and C_Y and returns these pairs in lieu of the actual values. In information theory, a MI I_{XY} of two random variables X and Y is formally defined as $I_{XY} = \sum_{i \in O(X)} \sum_{j \in O(Y)} p_{X_i \wedge Y_j} \log{(p_{X_i \wedge Y_j} / p_{X_i} p_{Y_j})}$, where p_{X_i}, p_{Y_j} are the marginal probability distribution functions of X and Y, respectively, and p_{X_i, Y_j} is the joint probability distribution function of X and Y. Intuitively, MI measures how much knowing one of these variables reduces the uncertainty about the other. Furthermore, the *entropy coefficient* (EC), using MI, measures the percentage reduction in uncertainty in predicting the dependent variable based on knowledge of the independent variable. When it is zero, the independent variable is of no help in predicting the dependent variable; and when it is one, there is a full dependency. The EC is directional and $EC(X|Y)$ for predicting the variable X with respect to variable Y is defined as I_{XY}/E_Y, where E_Y is the entropy of variable Y, formally $\sum_{Y_j} p_{Y_j} \log 1/p_{Y_j} = -\sum_{Y_j} p_{Y_j} \log p_{Y_j}$. Because I_{XY} also can be expressed as $E_X + E_Y - E_{XY}$ which has a easier form to compute, I choose this form to compute I_{XY}.

Algorithm 6 $Runtime\_Pruning(s, \alpha, \beta, E, C)$, s is a candidate VGFD $X \rightarrow Y$; α is the confidence threshold for a VGFD; β is the sampling size as a percentage; E is a set of triples. C is a set of cluster sets for each property.

```
1: $I \leftarrow Sampling\_Subjects(s, \beta, E)$          //Sampled subjects shared by the LHS and RHS.
2: $\{(X_i, Y_i)\} \leftarrow Distribution(s, I, E, C)$    //A list of value pairs where each pair consists of two single sampled values of LHS and RHS for the same subject.
3: $E_X = -\sum_{distinct\ x \in \{X_i\}} \frac{|\{X_i|X_i=x\}|}{|\{X_i\}|} \cdot log \frac{|\{X_i|X_i=x\}|}{|\{X_i\}|}$
4: $E_Y = -\sum_{distinct\ y \in \{Y_i\}} \frac{|\{Y_i|Y_i=y\}|}{|\{Y_i\}|} \cdot log \frac{|\{Y_i|Y_k=i\}|}{|\{Y_i\}|}$
5: $E_{XY} = -\sum_{distinct\ (x,y) \in \{(X_i,Y_i)\}} \frac{|\{(X_i,Y_i)|X_i=x \wedge Y_i=y\}|}{|\{(X_i,Y_i)\}|} \cdot log \frac{|\{(X_i,Y_i)|X_i=x \wedge Y_i=y\}|}{|\{(X_i,Y_i)\}|}$
6: **if** $(E_X + E_Y - E_{XY})/E_X < \alpha - 0.2$ **then**
7:     **return** TRUE
8: **return** FALSE
```

Paradies et al. [78] also used entropy to estimate the dependency between two columns in databases. Since they want to determine attribute pairs that can be estimated with high certainty, i.e. focusing on precision of the positives, they need a complex statistical estimator. In contrast, my aim is a fast filter that is good enough to remove most negatives, i.e. independent pairs, thus a statistical estimator is not necessary. I can avoid missing positives by setting a low enough threshold. In my experiments, the difference between EC for a 20% sample and EC of full data is less than 0.15 on average and the estimated values typically have higher ECs. For example, it is very rare that a VGFD estimated lower than 0.7 has an actual value above 0.9. Therefore, a threshold of 0.2 less than α (line 6) is a reasonable lower bound for filtering out independent pairs.

It is worthwhile to consider the relationship of Perplexity to this algorithm. The perplexity of a discrete probability distribution p is defined as $2^{H(p)} = 2^{-\sum_x p(x) \log_2 p(x)}$ where $H(p)$ is the entropy of the distribution and x ranges over events. The perplexity measures the uncertainty of a probability model. For example, given a model of a fair k-sided die (a uniform distribution over k discrete events), its perplexity is k. The larger the perplexity value, the more uncertain a probability model is. Thus it can be used to compare two probability models by checking if they have similar uncertainties, i.e. the perplexities. However, in my work, a mechanism is needed to compare how likely it is that the value distribution of two probability models are correlated. In other words, two probability models that have a similar level of uncertainty but are not correlated need to be ruled out. Therefore perplexity alone is insufficient for our problem.

7.5 Clustering Property Values

As introduced in the beginning of Section 7, it is necessary to cluster property values in order to discover dependencies with deeper semantics that allow for rounding errors, measurement errors, and distributions of values. For object property values, clustering groups all identifiers that stand for the same real world object by computing the transitive closure of sameAs. Clustering of literal values of datatype properties is more complete and will be discussed in the rest of this section. This is used to determine Dependency Equality (Definition 23) between two objects.

7.5.1 Pre-clustering

The pre-clustering process is a light-weight computation that provides two benefits for finer clustering later: it finds the minimum number of clusters and reserves expensive distance calculations for pairs of points within the same pre-cluster. Since the pre-clustering is used for VGFD discovery, there are three thoughts. First, the values to be clustered are from various properties and have very different features. So the clustering process needs to be generic in two aspects: (1) a pair-wise distance metric that is more general than linear ordering for different types of values and multiple feature dimensions, and (2) suitable for the most common distribution in real world, i.e. the normal distribution. Second, we prefer a comparatively larger number of clusters where elements are really close (if not, they may not be clustered). The reason is that the clusters will be used as class types for detecting dependencies. Larger values of k generate finer-grained class types, which in turn allow us to generate more precise VGFDs, albeit at the risk of bluring boundaries between classes and making it harder to discover some dependencies. This point also makes our approach different from many other pre-clustering approaches, e.g. [69], because their pre-clustering does not create true partitions of the values and their rigid clustering later could merge these groups into fewer clusters.

Based on the above thoughts, specifically, given a list of values, the pre-clustering process first selects a value that is closest to the center (I choose the mean for numeric values and discuss strings in the next paragraph), and then moves it from the list to be the centroid of a new group. Second, for each value on the list, if the distance to this centroid is within the threshold (I use the standard deviation), it will be moved from the list to the new group. Finally, the above process is repeated if the list is not empty. Thus the process generally finds the cluster around the original center first, and then the clusters further away from the center. This is much better than random selection, because if an outlier is selected, then most instances remain on the list for clustering after this round of computation.

To compute the center and distance of string values, we compute the weight of each token in a string according to its frequency in values for the property. Then we pick the string that has the largest sum of weights divided by the number of tokens in it as the center. The distance between two strings is the sum of weights of the different tokens in them. The intuition is that by taking these strings as a class, the most representative one is the one with the most common words. For example, the property *color* in DBpedia has values “light green”, “lime green”, etc. Then, the representative of these two strings is the common word “green”. For “light green”, the distance to ”lime green” will be less than that to “light red”, since ’‘red” and “green” are more common and have larger weights.

7.5.2 Optimal k-Means Clustering

There are several popular clustering methods, e.g. k-Means, Greedy Agglomerative Clustering, etc. However most of them need a parameter for the number of resulting clusters. To automatically find the most natural/best clusters, we designed the following unsupervised method of finding optimal clusters.

Algorithm 7 $Optimal\_kMeans(L, groups)$, L is a set of literal values; $groups$ is a set of pre-clustered groups of L.

```
$k = |groups|$
$oldGap = Gap\_Statistic(groups)$
$tmpC \leftarrow groups$
repeat
    $k = k + 1$, $C \leftarrow tmpC$, $tmpC \leftarrow \emptyset$                //$tmpC$ is the set of k clusters
    for each $i \leq k$ do
        $Init(m_i), c_i \leftarrow c_i \cup \{m_i\}, tmpC \leftarrow tmpC \cup \{c_i\}$   //$m_i$ is the center of each cluster
    repeat
        for each $x \in L$ do
            $i = \arg\min_i Distance(x, m_i)$
            $c_i \leftarrow c_i \cup \{x\}$
        for each $i \leq k$ do
            $m_i = Mean(c_i)$
    until $\forall i \leq k, m_i$ converges
    $newGap = Gap\_Statistic(tmpC)$
until $newGap < oldGap$
return $C$
```

The approach is inspired by the gap statistic [92] which is used to cluster numeric values with a gradually increasingly number of clusters. The idea is that when we increase k to above the optimum, e.g. adding a cluster center in the middle of an already ideal cluster, the pooled within-cluster sum of squares around the cluster mean decreases more slowly than its expected rate. Thus the gap between the expectation and actual improvement over different k will have a shape with an inflexion which indicates the best k. My approach improves upon this idea by leveraging preclusters in three ways: the algorithm starts at the number of pre-clusters instead of 1; in each round of k-Means, the initial centroids are selected according to pre-clusters; and the distance computation is only made among points within the same pre-cluster as opposed to between any pair.

The *Optimal_kMeans* algorithm is presented as Algorithm 7. At first, k is set to the number of pre-clusters. At each iteration, the algorithm increments k and selects k random estimated centroids m_i, each of which starts a new cluster c_i. The $Init()$ function selects the centroids from the pre-clusters in proportion to their sizes, i.e. each pre-cluster *group* has $k * |group|/|all\ values|$ random selected centroids. In each inner loop (line 8-13), every value is labeled as a member of the cluster whose centroid has the shortest distance to this instance among all centroids that are within the same pre-cluster as that value (line 10). Then each centroid is recomputed based on the cheap distance metric used in pre-clustering until the centroid does not change. Since the clustering is used to detect data in which string values might be abnormal due to typos or data conversion errors, I use edit distance as the distance metric for string values as opposed to the above pre-clustering. After each round of modified k-Means clustering, the algorithm computes the difference on $Gap(k)$ and stops the process if it is an inflexion point.

7.6 Experiments

In the experiments, I selected the SWRC, DBpedia and RKB[1] [38] data sets. All of them are widely used subsets of Linked Data that cover different domains. Experiments were conducted on a Sun workstation with 8 Xeon 2.93G cores and 6G memory. I observed that there are few dependencies with an LHS size larger than four and that such dependencies tend to have less plausible meanings. For this reason, I set the maximal size of a VGFD to four in the experiments. As I discussed at the end of Chapter 2, there are no existing systems with functions closely similar to

[1] http://www.rkbexplorer.com/data/

Table 7.2: System overall performance on SWRC, DBpedia and RKB data sets.

	SWRC	DBpedia	RKB
Number of Triples (M) / Properties	0.07 / 112	10 / 1114	38 / 54
Discovered VGFDs on Level 1	12	228	6
Discovered VGFDs on Level 2	37	304	3
Discovered VGFDs on Level 3	2	126	0
Discovered VGFDs on Level 4	0	53	0
Total discovered VGFDs	51	721	9
Time for Clustering (s)	18	114	396
Time for Level 1 (s)	11	172	67
Time for Level 2 (s)	20	246	44
Time for Level 3 (s)	4	108	0
Time for Level 4 (s)	1	47	0
Total Time (s)	54	687	507
Reported Abnormal Triples	75	2868	227

this system. So I did not compare this system with others from a whole system perspective. However, to validate the system, I compared each of system sub-function with other algorithms that provide the same function.

In the first experiment, I compared the overall performance of the system on three data sets. The sampling size β used in runtime pruning is 20%. In Table 7.2, it can be seen that the running time appears to be more heavily influenced by the number of properties than the data set size. Note that RKB has more triples but fewer properties than DBpedia, and thus has more triples per property. This leads to a longer clustering time, but thanks to static and runtime pruning, the total time to find VGFDs is less.

Table 7.3 gives some VGFDs from the three data sets and their short descriptions. I listed them into three groups: VGFDs with size 1 and size 2, and VGFDs based on clusters. For example, the last VGFD based on clusters means that a school's type determines the range of upper age, because we have the clusters shown in Table 7.4. In DBpedia, among 200 samples out of 2868 abnormal triples, 173 of them (86.5%) are confirmed to be true errors in the original data. The correctness of 10 of the remaining triples was difficult to judge. SWRC

Table 7.3: Some VGFDs from the three data sets. The first and second group of VGFDs are of size 1 and 2 respectively. The third group is a set of VGFDs with clustered values.

VGFD and its Description
genus→family Organisms in the same genus also have the same family.
writer→genre A work's writer determines the work's genre.
teamOwner→chairman The teams with the same owner also have the same chairman.
composer→mediaType The works by the same composer have the same media type.
militaryRank→title The people of the same military rank also have the same title.
location→nearestCity The things at the same location have the same nearest city.
topic→primaryTopic The papers with the same topic have the same primary topic.
manufacturer+oilSystem →compressionRatio The manufacturer and oil system determine the engine's compression ratio.
publisher ∘ country →language The publisher's country determines the language of that published work.
article-of-journal+has-volume→has_date A journal's volume number determines the date of publications in this journal.
faculty→budget The size of the faculty determines the budget range.
militaryRank→salary The military rank determines the range of salary.
occupation→salary The occupation determines the range of salary.
type→upperAge A school's type determines the range of upper age.

School Type	Upper Age
Elementary School	{11, 12}
Secondary School	{13, 14}
High School	{18, 19}

Table 7.4: Correlation between values of school type and clusters of property upper age.

and RKB have 51% and 62% precision respectively. I believe the lower precision for SWRC is because it has a higher initial data quality and its properties have a much smaller set of possible values than those of DBpedia. I list a number of confirmed erroneous triples in Table 7.5, where r, o, i, p, s are prefixes for `http://www.dbpedia.org/resource/`, `http://www.dbpedia.org/ontology/`, `http://acm.rkbexplorer.com/id/`, `http://www.aktors.org/ontology/portal/` and `http://data.semanticweb.org/`. These errors are listed in two groups: the first is outliers and the other is VGFD violations. For example, the first triple in the first group is reported as an outlier after automatic clustering. The first triple in the second group violates the VGFD that a journal's volume number determines the date of publications in this journal, because the triple's subject is an article published in certain issue of a journal while its publish date is not in the cluster of values for the articles published in the same issue of journal.

Next, to check the impact of our pruning algorithms, I performed an ablation study using DBpedia that removes these steps. Table 7.6 shows that using static and runtime pruning respectively saves over 62% and 55% of time compared to using neither. Because they utilize different characteristics, using them together saves 85% over neither. When not pruning, the few additional VGFDs discovered lead to fewer abnormal triples than those discovered with pruning (on average 2.2 per VGFD vs. 3.97 per VGFD). Thus the pruning techniques not only save time but do not affect the abnormality detection much.

Besides pruning, I also checked the impact of our pre-clustering. Because my approach is based on a generic pair-wise distance, I wanted to compare it with a simpler one based on the linear ordering of values where the distance is just the

Table 7.5: Some confirmed erroneous triples in the three data sets. The first group is outliers and the second group is VGFD violations.

1	<r:Shanghai_Jiao_Tong_University, o:university/undergrad, 194323445>
2	<r:Harrow_College, o:School/upperAge, 2009.0>
3	<r:Melbourne_Grammar_School, o:School/ranking, 2006.0>
4	<r:Dembela, o:Place/coordinates, coord|N|W>
5	<r:Hutt_Valley_High_School, o:EducationalInstitution/principal, r:2008>
6	<r:Wake_Island, o:Island/country, r:United_States_Air_Force>
7	<r:Albuquerque_Plaza, o:Building/floorCount, 2221>

8	<i:journals/jair/DarwicheP97, p:has-date, 1998>
9	<r:Wiktionary, o:Work/language, r:History_and_development>
10	<r:varedo, o:City/province, r:Province_of_Milan>
11	<i:796511, p:has-date, to-10-01>
12	<r:Google_Maps, o:Work/language, r:Coverage_details_of_Google_Maps>
13	<s:person/bastian-quilitz, s:ns/swc/ontology#affiliation, research assistant>
14	<s:person/ulf-leser, s:ns/swc/ontology#affiliation, professor>

Table 7.6: The impact of my pruning techniques.

	None	Static	Runtime	Both
Time (s)	4047	1529	1817	687
VGFDs	746	741	729	721
Abnormal	2923	2915	2887	2868

difference between numbers. After each iteration of clustering around the mean, this alternative, referred to as SortSeq, recursively clusters on two remaining value sets: one is above the mean and the other below the mean. To handle strings in this approach, I sort them alphabetically and assign each a sequence number. Another baseline, LetSum, gives each letter l_i ($i \in \{1...26\}$) a value based on alphabetic ordering and assigns each string s the value $\sum_{l_i \in s} l_i * (1/27)^i$. It is analogy of how fraction number is represented. In base 10, the number 0.312 is equivalent to: $3(1/10)+1(1/10)^2+2(1/10)^3$. Similarly, for example, the string 'cab' is represented as $3(1/27)+1(1/27)^2+2(1/27)^3$, assuming each letter is represented as a number starting from 1. Table 7.7 shows that VGFDs and abnormal data that are based on the baseline clustering are both less than that of our approach. Among the VGFDs

Table 7.7: Comparison between preclustering with an alternative called SortSeq on VGFDs using the clusters and abnormal data found based on these VGFDs.

	Preclustering	SortSeq	LetSum
Time (s)	114	83	89
VGFDs	42	23	53
Abnormal	625	391	234

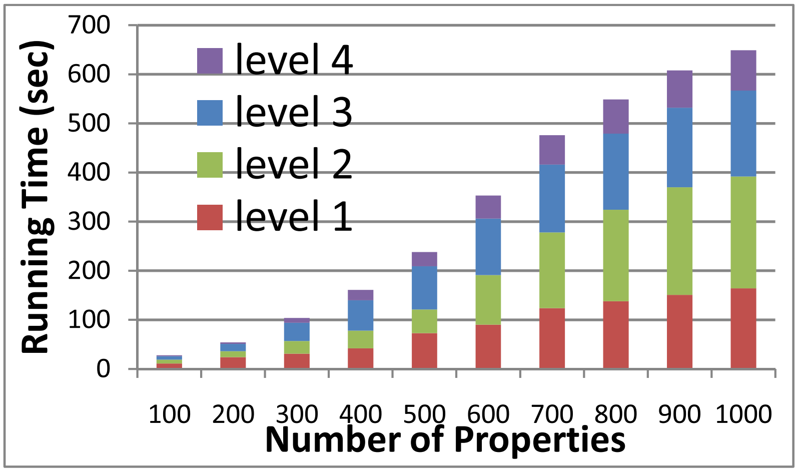


Figure 7.8: The effect of number of properties on the VGFD searching time.

not found by the SortSeq, most are for string values. SortSeq finds fewer VGFDs and less abnormal data, because it naively assumes that the more common leading characters two strings have, the more similar they are. LetSum tends to cluster values more evenly and creates fewer clusters. The smaller number of clusters means larger clusters, which leads to more likely dependencies and thus more discovered VGFDs. But it does not capture the really majority and minority of the values and so the detected abnormal triples are much less. Thus, my pre-clustering using cheap and generic computation captures the characteristics of different property values. Besides the comparison between our pre-clustering with other alternatives, we also tested the necessity of our pre-clustering by running the system without pre-clustering step. Specifically, we input the values of each property as one group into opti-kmeans clustering. Then the *Optimal_kMeans* clustering would start with number of one to automatically find the optimal number of clusters. The result shows

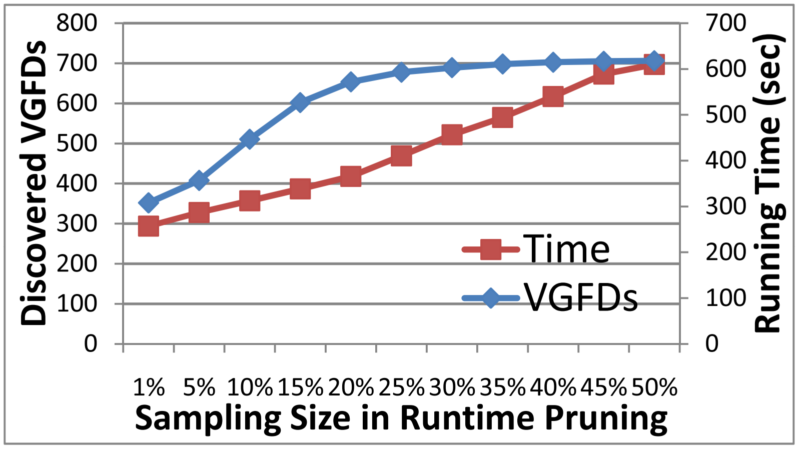


Figure 7.9: The effect of sampling size in runtime pruning on the VGFD searching time.

that without pre-clustering, the time cost on *Optimal_kMeans* clustering on DBpedia is 359 seconds, while the total time using pre-clustering and *Optimal_kMeans* is only 114 seconds. The result demonstrates that the pre-clustering step give much benefit for later finer-grained clustering step. Appendix A gives an example comparison between results of our proposed clustering and results of the LetSum clustering.

Knowing that pre-clustering and pruning are useful for the system, I systematically checked the trend of system performance, especially time, by using these techniques. To be comparable on data set size, I picked subsets of properties from DBpedia. For each size, I randomly draw 10 different groups of this size and average the time over 10 runs. Fig. 7.8 shows that the time for every level of the VGFD search almost follows a linear trend.

Fig. 7.9 shows the effect of sampling size β used in runtime pruning on the system. It can be seen that the running time is in linear proportion to the sampling size. As the VGFD curve shows, $\beta = 0.2$ is sufficient to find most dependencies for DBpedia.

Both Fig. 7.8 and Fig. 7.9 give some idea of how the system scales. Fig. 7.8 shows when the number of properties increases, how the time cost of the system varies. I believe the number of properties usually are expected in proportional of data set size, especially when these properties are randomly picked from the same original data sets. For example, a data set with twice of the number of properties is expected to have twice of the number of triples. If it is, then this figure indirectly shows how the system scales on different sizes of data. Fig. 7.9 can also be seen as how the system scales from another perspective. Because when the sampling size increases, I think it can roughly be seen as keeping the same sampling size on larger data sets. For example, when we double the sampling size, the time cost can be roughly expected to be similar to using the same sampling size on a data set with double the number of triples. Therefore, when we increase the sampling size, this figure can also be interpreted as increasing the data set size while keeping the same sampling size.

Chapter 8

Conclusion

In this final chapter, I conclude this thesis with analysis of algorithms designed in this work and future work that can improve on it.

8.1 Analysis

To help improve Semantic Web data quality, I proposed and implemented several approaches for detecting abnormal Semantic Web data. To this end, I divided real world Semantic Web data into three scenarios for dealing with object property triples. Based on these systems, my final system implements a more general mechanism. Each diagram of these systems in this thesis have been shown in previous chapters and is put in dotted boxes in Fig. 8.1 for comparison. To conclude this thesis, I make a short summary and analysis of each system in the following.

To deal with first type of scenario, the system is based on discovering characteristics from a training data set first and then comparing between the training dataset and the data being investigated. The training dataset has to satisfy two requirements. First it is generally correct. This requirement means that the data set could have few errors but no consistent repeated errors, i.e. no systematic errors. Although it is hard to quantify the percentage of errors that is allowed, based on my experiments, when the erroneous data is less than 5%, it usually would not affect the general patterns in the data set. The second requirement to the data set is that

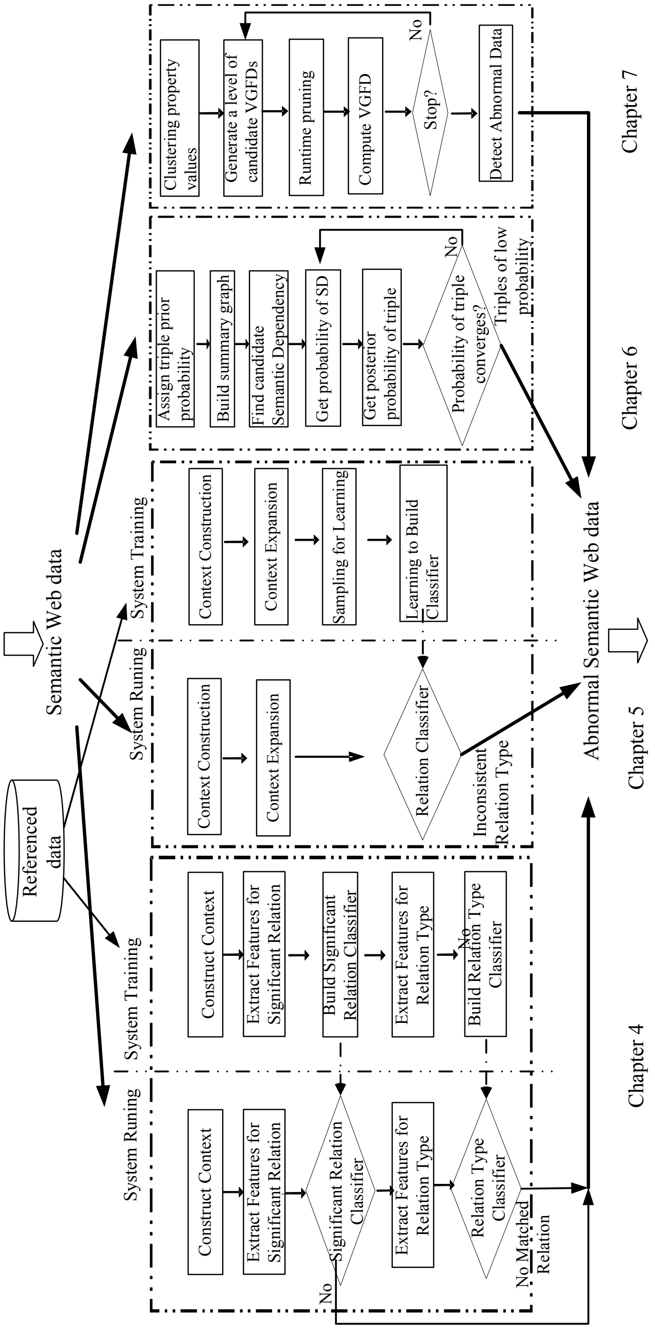

Referenced data
Semantic Web data
System Runing
System Training
Construct Context
Extract Features for Significant Relation
Build Significant Relation Classifier
Extract Features for Relation Type
Build Relation Type Classifier
Significant Relation Classifier
Relation Type Classifier
No
No Matched Relation
Chapter 4
Context Construction
Context Expansion
Sampling for Learning
Learning to Build Classifier
Relation Classifier
Inconsistent Relation Type
Chapter 5
Assign triple prior probability
Build summary graph
Find candidate Semantic Dependency
Get probability of SD
Get posterior probability of triple
Probability of triple converges?
Triples of low probability
Chapter 6
Clustering property values
Generate a level of candidate VGFDs
Runtime pruning
Compute VGFD
Stop?
Detect Abnormal Data
Chapter 7
Abnormal Semantic Web data

Figure 8.1: The architecture of whole system. The four dashed boxes are parallel algorithms for each situation.

it is comprehensively described with respect to the vocabulary of the ontologies, i.e. the data gives much contextual information for each triple. The comprehensiveness is important in that it is an approximate interpretation of the closed world assumption: given the ontology vocabularies, most of objects in the domain which the dataset describes have rich usages of all available concepts and properties. The system utilizes the conditions in this scenario and consists of two components. The first component extracts several quantitative metrics from the context of a triple and uses them to check if there is a credible, significant relation between two instances in this triple. The other component extracts patterns from the context of this triple and matches them against patterns learned from each type of relation in the training data set. Because the system makes stronger assumptions than that of other systems, it is the most efficient both on time (4 hours on SWRC) and space.

The open world assumption is more often applied to Semantic Web data. In the second scenario, the training data and the data to be evaluated are still assumed generally correct. Corresponding to this scenario, the system is integrated in one step as opposed to two steps in the previous system and it exploits a new classifier that is different from those that the previous system used. The system I developed utilizes a vector space model to represent and support expansion of the context of each triple. It incorporates a learning model that does not treat missing triples as negative examples in order to not make the closed world assumption. The system is good at dealing with data sets that are not fully described for every triple. This is accomplished by expanding the contexts and the vector space model of contexts costs more memory than any of the other systems that deals with object property triples. Although the learning model that considers the open world assumption is complex, by using sampling during learning, the learning can still be finished under reasonable amount of time, e.g. 5 hours for SWRC.

The third situation is that there is no clean training data set available. Without a training data set to learn from, the third system improves the patterns that are discovered from the data to be evaluated and are similar to what is learned from the training data in the previous systems. The first improvement considers that the data contributing to the patterns have different truth probabilities. The second improvement considers the logical consistency among these patterns. Then the system iteratively adjusts the truth probabilities of triples based on the patterns supporting them. Besides improving system accuracy, the system also improves on the pattern discovery process and the operation to get instantiations of each pattern by generate a summary of original RDF graph. Thus when using an appropriate stopping threshold and initial prior probability of triples, the time is even less than the previous system dealing with data set to which the open world assumption is applied (e.g. 3 hours on SWRC).

All of the previous systems essentially use typical patterns to cast a majority vote to determine if a triple is abnormal, i.e. if the context of the triple lacks sufficient patterns. Many patterns are learned from part of the data set, i.e. some patterns are valid for a fragment of the data, and do not apply generally. Although there are weighting schemes to adjust their effect, some of them are relatively noisy patterns and might be spurious. However, if the system only focuses on strong patterns, it can use them to detect abnormal data by checking conflict with a single pattern as opposed to using a majority of patterns to vote for the regularity of the triple. Further it will be clearer and easier to explain to the system user why some data is abnormal. Another common drawback of previous systems is that the patterns are based on sharing the same values between the LHS property and RHS property. An immediate effect of this drawback is that not many patterns are based on datatype properties, because the values of datatype properties are not instances and then different properties cannot be connected on them in RDF graphs. Therefore it is necessary to discover patterns that do not require value reuse. Having the above thoughts, to find more implicit, stronger patterns, the last approach extended the concept of functional dependency in databases into value-clustered graph functional dependencies. These dependencies are devised to capture implicit correlations among all types property (both object property and datatype property) values, even if there is no explicit connection between these values (i.e. triples may not be connected through reused values). The extension includes several aspects. First, it introduces operators to combine properties in order to retrieve more semantics about the instances, such as composition and conjunction operators that chain properties together and consider collections of property chains, respectively. Second, it considers property values with similar semantics instead of only based on syntactic comparison.

All of these algorithms are designed for different situations and can be easily applied on various Semantic Web data. To sum up, this thesis has the following contributions.

1. I have developed three algorithms to evaluate the data quality issues of object property triples in different situations according to completeness and entire quality of data. Demonstrated through experiments, given a well described training data set, the first algorithm can achieve over 80% F-score on classifying normal data and abnormal data. Given a data set to which the open world assumption is applied (e.g. 9% original data is removed from the data set), the F-score of second algorithm is 8% better and drops less than 3% compared to when using a complete data set to learn from. When not given a clean data set for training, the third algorithm extracted the patterns similar to those in previous algorithms while improving accuracy of their probabilities by taking into account truth probabilities of triples. Shown in experiments, when given data sets containing 9% or more incorrect data, the third system can get better performance than that of previous systems using these given data sets.

2. I have demonstrated what kind of context is useful for evaluating object property triples in order to increase accuracy of data quality problem. The context for a triple essentially are paths on RDF graphs connecting the pair of objects in this triple. Given the context, the system can appropriately retrieve the semantics representing their relationship from these paths by interpreting the sequence of predicates on it. Further combining and comparing semantics on different paths, the system can determine the direct relationship between the pair of objects with certain probabilities. Based on this idea, the system can report how likely the triple is abnormal through how similar the triple is to that the system determined. All the experiments for the algorithms in

this thesis essentially show the effectiveness of these contexts, though they might be in different forms, such as RDF subgraphs, semantic dependencies or VGFDs.

3. I have extended the concept of functional dependency from relational databases into RDF graphs and used them to detect abnormal Semantic Web data. The experiments on three real world data sets show that the algorithm has a decent precision (e.g. 86% out of 2868 reported errors on DBpedia) on detecting errors in original data sets. Meanwhile, results demonstrated that the system can detect useful and meaningful dependencies based on both syntactics and semantics (through clustering). Futhermore, several pruning techniques in the algorithm make the system applicable for large scale real world Semantic Web data set, since it only takes 12 minutes on a data set with 10 million triples using servers with 8 Xeon 2.93G cores.

8.2 Future Work

Although I have conducted many experiments to test the approaches and algorithms, there is much room to improve these systems. Some of the key points that can be improved are discussed below.

First, in Chapter 6, I used an iterative approach adjusting the triple's probability according to the probability of semantic dependencies that can support it. There is a theoretical question if this iterative process is guaranteed to converge or stop at certain specified conditions. Although I gave a brief analysis on how the system is designed to be guaranteed to stop in any conditions, it would be ideal if there is formal proof. Because the iterative process consists of several steps of computation, the formal proof requires the analysis of the input and output of each step. The analysis must consider what is changed by the computation in each step and how each change happens. From this analysis, I could clearly see how each step can affect the next step. Therefore, given the initial input before all steps, the analysis results will show what are changed after each step in every iteration and so that I can conclude if this iterative process can make the triple's truth probability converge, given the input stopping threshold. Although the analysis mainly depends on the actual data set, it is possible to conduct the analysis on some extreme cases to bound the possible changes and the expected changes based on my observations of the system running on several different real world data sets. Besides that, since some steps of the probability computation are inspired from classic probability theory, it is ideal that I can use analysis techniques or results from existing similar probability theories that can apply in our problem, e.g. naive Bayesian.

Second, to better understand and improve the approach using value-clustered functional dependency, I think it is better to have a theoretical complexity analysis. Because this approach is extended from functional dependency in databases and it is designed to deal with RDF data or, more generally, the data in graph models, it is likely to be more computationally complex than discovering functional dependency in databases. Thus it would be valuable to give some theoretical analysis and comparison with representative approaches for discovering functional dependencies in databases. The comparison would help us to better understand the strength of our approach and if the designed approach is the best choice compared to approaches for discovering simpler functional dependencies in databases. Before detailed complexity analysis, a possible future work is to report whether the heuristics or pruning methods would not be effective in some situations and, if it is, what these cases are and how likely these cases are. After that, it is appropriate to analyze the complexity of the system in two situations: the worst case and the expected case. The worst case would be when the heuristics/pruning fail, while the expected case is when they provide some benefit. Besides the complexity analysis, I also plan to analyze how to improve the VGFD's capability on detecting abnormal data. To this end, I need to analyze in detail what kind of VGFDs can detect more abnormal data than others and why they do. Following it, the question I can try to answer is how to make other VGFDs that improve on this capability or how to find more such VGFDs.

Besides the works on theoretical aspects, I recognize that the RHS of VGFDs discovered by current system are only single original properties. To make it more general, I need extend it to composite properties as used in the LHS of VGFDs.

I believe that this extension will make VGFDs more theoretically complete. But it would greatly increase the complexity. Therefore, I need to carefully consider only those situations where it is necessary to extend the RHS and if it can bring any meaningful VGFDs. Then I also need to consider how to improve the new computation process, e.g. some additional heuristics or pruning techniques. The experiments could be essentially similar to the process that I conducted on current system. I input the system different popular data sets that have different sizes and are on different domains. Then I record detailed performance of the system on each step. It is also possible that I pick subsets of each data set with increasing sizes of number of triples, properties, etc. Then based on these subsets, I can systematically analyze the performance trend of the system on each step. Besides the performance, more importantly I can investigate the quality of system results. I may ask several Semantic Web experts to manually verify a small portion of samples of results and then summarize them. Or I may use some crowd sourcing mechanism to public verify the results.

Although functional dependency is by far the most common form of integrity constraints in databases, multivalued dependency in databases is also an important and useful concept. If I can devise a way to extend the multivalued dependency to apply to RDF graphs, it might bring more potential integrity constraints in RDF data and so be able to discover more erroneous triples. To accomplish this goal, the first of future steps is to review and critically think about previous approaches for discovering multivalued dependencies in databases.

This thesis discussed the problem of low quality data on the Semantic Web, previous research on similar problems and several algorithms that I designed to help detect such quality issues in real world Semantic Web data. The thesis can help researchers notice that the significance of data quality issues on the Semantic Web. Further it can help people be familiar with previous approaches and thoughts on this problem. Finally, the results of the systems and experiments that I developed can help researchers better create, consume Semantic Web data. Finally, it promises to lead Semantic Web data (e.g. Linked Data) to a higher quality. These higher quality of Semantic Web data then might save the huge cost of low quality data (e.g. 600 billion dollars spent each year discussed at the beginning of this thesis), avoid negative impacts on people's everyday life (e.g. the potential health impacts of radiation safety tests discussed at the beginning of this thesis) and more.

Appendix A

Example comparisons of clustering results

This section gives detailed comparison on results of clustering algorithms between the one we proposed and the baseline LetSum which is discussed in Section 7.6. The first example is clustering on values of property `http://dbpedia.org/ontology/SoccerClub/managerTitle`. The following is the list of clusters proposed by our system.

1. Manager, First Team Manager, Senior Mens Manager, Vice-President, Player-manager, Director of Football, Player/Manager, Club Secretaries, Head Coach Pasi Rautio manager = Juhani Vesanen, Caretaker manager, Trainer-manager, Head Coach Teemu Ryypp?manager = Antti Korpela, 1st Team Manager, Technical Director<br>/Manager, Interim Reserve Manager, Last Manager, Player manager, Head Coach Pavel Tresnak manager = Oiva Tapio, Manager-Captain, Team Manager, Team manager

2. Coach, First Team Coach, Head coach and Director of Operations, Senior Coach, Acting head coach, Reserve team coach, Chief Coach, 1st Team Coach, Interim Head Coach, First Grade Coach, Player-coach, Team head, Head Coach, Head coach,

3. CEO,

4. Teamchef,

5. Director, Sports Director, Technical Director, Director general,

6. Managers, Joint Managers, Caretaker Managers, Joint managers, Co-managers, Co-Managers,

7. President,

8. Captain,

9. Trainer,

10. Co-Trainers,

11. Coaches,

12. D.T, D.T.,

13. Secretary,

14. Founder,

The second list is the clusters suggested by LetSum clustering. Each string is also associated with its number representation used in clustering.

1. Joint Managers, Joint managers,
   Interim Head Coach, Last Manager,
   Interim Reserve Manager,

2. Head Coach, Head coach, Founder, First Team Coach, First Team Manager, Head coach and Director of Operations, Head Coach Pasi Rautio manager = Juhani Vesanen,
   Head Coach Teemu Ryypp?manager = Antti Korpela, First Grade Coach, Head Coach Pavel Tresnak manager = Oiva Tapio,

3. Manager, Managers, Manager-Captain,

4. President, Player-manager, Player/Manager,
   Reserve team coach, Player manager, Player-coach,

5. Coach, Captain, CEO, D.T., Director, Director of Football, D.T, Co-managers, Club Secretaries, Caretaker Managers, Chief Coach, Co-Trainers, Caretaker manager, Coaches, Co-Managers, Director general,

6. Secretary, Senior Mens Manager, Teamchef, Trainer, Senior Coach, Sports Director,
   Team Manager, Team manager, Technical Director,
   Trainer-manager, Team head,
   Technical Director<br>/Manager,

7. Acting head coach, 1st Team Coach, 1st Team Manager,

8. Vice-President,

The second example is one property `http://dbpedia.org/ontology/Rocket/function`. The following is the results of our system.

1. Re-usable orbital launch vehicle, Man-rated re-usable orbital launch vehicle, All-solid small orbital launch vehicle, Manned partially re-usable launch and reentry system, Unmanned Launch Vehicle, Technology demonstrator for liquid propulsion based VTOL rocket flight,
   Prototype ICBM<br>Expendable launch system, LEO launch vehicle, Manned/unmanned LEO and Lunar launch vehicle, Space Station launch vehicle, Satellite launch vehicle, Mid-Heavy Lift Launch System, Super Heavy launch vehicle, Small, modular component launch vehicle, Cargo Launch Vehicle (unmanned),
   EELV/Medium-heavy launch vehicle, Vehicle for re-entry studies, Sub-Orbital Test Vehicle

2. Expendable launch system, Medium Lift Launch System, High Expendable launch system<br>Sounding rocket, Heavy suborbital launch system,
3. SLBM,
4. Interim carrier rocket, GTO Carrier rocket, Manned heavy-lift multi-purpose carrier rocket,
5. Sounding rocket,
6. ASAT booster,
7. Unmanned test capsule,
8. Unmanned reusable spaceplane technology demonstrator,
9. ICBM,
10. Prototype ICBM,
11. A-1: Experimental,
12. Anti-satellite weapon,
13. Intercontinental ballistic missile,

Then the list of clusters suggested by LetSum is given below.

1. Medium Lift Launch System, Man-rated re-usable orbital launch vehicle, Manned heavy-lift multi-purpose carrier rocket,
   Medium expendable Launch vehicle, Launch System, Medium carrier rocket, Manned partially re-usable launch and reentry system, Manned Re-usable orbital launch vehicle,
   Medium expendable launch system, Manned launch vehicle, Man-rated orbital launch vehicle, LEO launch vehicle, Manned LEO launch vehicle,
   Manned/unmanned LEO and Lunar launch vehicle, Manned LEO and Lunar launch vehicle, launch vehicle, Medium expendable Carrier rocket, Manned

lunar carrier rocket, Mid-Heavy Lift Launch System, Launch vehicle, Manned Re-usable Spaceplane, Man-rated LEO carrier rocket, Manned expendable launch system, man-rated orbital launch vehicle, Medium/Heavy launch vehicle, Medium launch vehicle, Manned launch system, Manned sub-orbital launch vehicle,

2. Orbital launch vehicle, Orbital carrier rocket,
   Prototype ICBM<br>Expendable launch system, Prototype ICBM, Prototype expendable launch system,

3. ICBM, Intercontinental ballistic missile, Interim carrier rocket, Heavy carrier rocket, High Expendable launch system<br>Sounding rocket, Heavy Manned Launch vehicle, ICBM/Launch vehicle, Heavy suborbital launch system, Heavy launch vehicle, GTO Carrier rocket, Heavy expendable launch system,

4. Sounding rocket, Re-usable orbital launch vehicle, Small orbital launch vehicle, Small carrier rocket, Space Station launch vehicle, Satellite launch vehicle, Suborbital launch system, Small expendable launch system, Sub-Orbital Test Vehicle,
   Small, modular component launch vehicle, Super Heavy launch vehicle, SLBM, Small launch vehicle,

5. Unmanned Launch Vehicle, Technology demonstrator for liquid propulsion based VTOL rocket flight, Unmanned launch vehicle, Unmanned LEO and Lunar launch vehicle, Unmanned reusable spaceplane technology demonstrator, Test vehicle, Unmanned Re-usable Spaceplane, Unmanned test capsule, Unmanned Re-usable Spaceplane technology demonstrator,

6. Expendable launch system, Expendable launch vehicle, Experimental carrier rocket, EELV/Medium-heavy launch vehicle,
   Expendable carrier rocket, Expendable launch system<br>Sounding rocket,

7. Vehicle for re-entry studies,

8. Carrier rocket, All-solid small orbital launch vehicle, Anti-satellite weapon, Cargo Launch Vehicle (unmanned), ASAT booster,

9. A-1: Experimental(0.037037052),

Bibliography

[1] Openlink software, sponger technology. *http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VirtSponger*.

[2] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *SOCIAL NETWORKS*, 25:211–230, 2001.

[3] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22:207–216, June 1993.

[4] Mike M. Ahlers. *TSA orders 're-tests' of radiation levels on airport body scanners*. CNN, http://edition.cnn.com/2011/US/03/11/tsa.body .scanners/index.html, 12 Mar. 2011.

[5] Pierre Allard, Sébastien Ferré, and Olivier Ridoux. Discovering functional dependencies and association rules by navigating in a lattice of olap views. In Marzena Kryszkiewicz and Sergei A. Obiedkov, editors, *CLA*, volume 672 of *CEUR Workshop Proceedings*, pages 199–210. CEUR-WS.org, 2010.

[6] F. J. Anscombe and I. Guttman. Rejection of outliers. *Technometrics*, 2(2):123–147, 1960.

[7] Kemafor Anyanwu, Angela Maduko, and Amit Sheth. Semrank: ranking complex relationship search results on the Semantic Web. In *WWW '05*, pages 117–127, New York, NY, USA, 2005. ACM.

[8] William W. Armstrong. Dependency Structures of Data Base Relationships. In *Proc.˜of IFIP World Computer Congress*, pages 580–583, 1974.

[9] Paolo Atzeni and Valeria De Antonellis. *Relational database theory*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1993.

[10] Yonatan Aumann and Yehuda Lindell. A statistical theory for quantitative association rules. In *Journal of Intelligent Information Systems*, pages 261–270, 1999.

[11] Kenneth Baclawski, Mieczyslaw M. Kokar, Paul A. Kogut, Lewis Hart, Jeffrey E. Smith, William S. Holmes, III, Jerzy Letkowski, and Michael L. Aronson. Extending uml to support ontology engineering for the semantic web. In *Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, &#171;UML&#187; '01, pages 342–360, London, UK, UK, 2001. Springer-Verlag.

[12] A. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3-4):590–614, August 2002.

[13] V. Barnett and Lewis T. Outliers in statistical data. 3rd edition. *Biometrical Journal*, 37(2):256–256, 1995.

[14] Catriel Beeri, Martin Dowd, Ronald Fagin, and Richard Statman. On the structure of armstrong relations for functional dependencies. *J. ACM*, 31:30–46, January 1984.

[15] Tim Berners-Lee. Sir tim berners-lee talks with talis about the semantic web. February 2008.

[16] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[17] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[18] Rudolf K. Bock and Werner Krischer. *The Data Analysis Briefbook*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1998.

[19] Philip Bohannon, Wenfei Fan, Michael Flaster, and Rajeev Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 143–154, New York, NY, USA, 2005. ACM.

[20] Janez Brank, Marko Grobelnik, and Dunja Mladeni04. A survey of ontology evaluation techniques. In *In In Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005*, 2005.

[21] Paul G. Brown and Peter J. Hass. Bhunt: automatic discovery of fuzzy algebraic constraints in relational data. In *Proceedings of the 29th international conference on Very large data bases - Volume 29*, VLDB '2003, pages 668–679. VLDB Endowment, 2003.

[22] Doina Caragea, Vikas Bahirwani, Waleed Aljandal, and William H. Hsu. Ontology-based link prediction in the livejournal social network. In Vadim Bulitko and J. Christopher Beck, editors, *SARA*. AAAI, 2009.

[23] Kei-Hoi Cheung, Kevin Y. Yip, Andrew Smith, Remko Deknikker, Andy Masiar, and Mark Gerstein. Yeasthub: a semantic web use case for integrating data in the life sciences domain. *Bioinformatics*, 21:85–96, January 2005.

[24] E. F. Codd. Relational completeness of data base sublanguages. In *Database Systems*, pages 65–98. Prentice-Hall, 1972.

[25] Gao Cong, Wenfei Fan, Floris Geerts, Xibei Jia, and Shuai Ma. Improving data quality: consistency and accuracy. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 315–326. VLDB Endowment, 2007.

[26] S Decker, S Melnik, F Van Harmelen, D Fensel, M Klein, J Broekstra, M Erdmann, and I Horrocks. The semantic web: The roles of xml and rdf. *IEEE Internet Computing*, 4(5):63–74, 2000.

[27] Wayne Eckerson. Data Quality and the Bottom Line: Achieving Business Success through a Commitment to High Quality Data. Technical report, The Data Warehousing Institute, 2002.

[28] Larry P. English. *Improving data warehouse and business information quality: methods for reducing costs and increasing profits*. John Wiley & Sons, Inc., New York, NY, USA, 1999.

[29] Eleazar Eskin. Anomaly detection over noisy data using learned probability distributions. In Pat Langley, editor, *ICML*, pages 255–262. Morgan Kaufmann, 2000.

[30] Victoria P. Evans. *Strategies for Detecting Outliers in Regression Analysis [microform] : An Introductory Primer / Victoria P. Evans*. Distributed by ERIC Clearinghouse, [Washington D.C.] :, 1999.

[31] Ronald Fagin. Functional Dependencies in a Relational Data Base and Propositional Logic. *Ibm Journal of Research and Development*, 21:543–544, 1977.

[32] Ronald Fagin and Moshe Y. Vardi. The theory of data dependencies - an overview. In Jan Paredaens, editor, *ICALP*, volume 172 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 1984.

[33] Christos Faloutsos, Kevin S. Mccurley, and Andrew Tomkins. Fast discovery of connection subgraphs. In *KDD ’04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 118–127, New York, NY, USA, 2004. ACM Press.

[34] Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Database Syst.*, 33:6:1–6:48, June 2008.

[35] Thomas Franz, Antje Schultz, Sergej Sizov, and Steffen Staab. Triplerank: Ranking semantic web data by tensor decomposition. In *International Semantic Web Conference*, pages 213–228, 2009.

[36] Christian Fürber and Martin Hepp. Using sparql and spin for data quality management on the semantic web. In Witold Abramowicz and Robert Tolksdorf, editors, *BIS*, volume 47 of *Lecture Notes in Business Information Processing*, pages 35–46. Springer, 2010.

[37] Lise Getoor and Christopher P. Diehl. Link mining: a survey. *SIGKDD Explor. Newsl.*, 7:3–12, December 2005.

[38] Hugh Glaser, Ian C. Millard, and Afraz Jaffri. Rkbexplorer.com: a knowledge driven infrastructure for linked data providers. In *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*, ESWC'08, pages 797–801, Berlin, Heidelberg, 2008. Springer-Verlag.

[39] F. E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11:1–21, 1969.

[40] Thomas R. Gruber and Patrice O. Gautier. Machine-generated explanations of engineering models: A compositional modeling approach. In *In Proc. International Joint Conference on Artificial Intelligence*, pages 1502–1508. Morgan Kaufmann, 1993.

[41] N. Guarino. *Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy.* IOS Press, Amsterdam, The Netherlands, The Netherlands, 1st edition, 1998.

[42] R. Guha, R. Mccool, and R. Fikes. Contexts for the semantic web. In *International Semantic Web Conference, volume 3298 of Lecture Notes in Computer Science*, pages 32–46. Springer, 2004.

[43] Peter J. Haas, Fabian Hueske, and Volker Markl. Detecting attribute dependencies from query feedback. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 830–841. VLDB Endowment, 2007.

[44] Olaf Hartig. *Provenance Information in the Web of Data*, pages 1–9. CEUR-WS, 2009.

[45] Sven Hartmann, Sebastian Link, and Markus Kirchberg. A subgraph-based approach towards functional dependencies for XML. In Nagib Callaos, William Lesso, Shahram Rahimi, Verra Boonjiing, Jihad Mohamad, Te-Kai Liu, and Klaus-Dieter Schewe, editors, *Computer Science and Engineering: II*, volume IX of *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI)*, pages 200–211. International Institute of Informatics and Systemics (IIIS), 2003.

[46] Douglas M. Hawkins. *Identification of Outliers*. Chapman and Hall, 1980.

[47] Martin Hepp. Goodrelations: An ontology for describing products and services offers on the web. In *Proceedings of the 16th international conference on Knowledge Engineering: Practice and Patterns*, EKAW '08, pages 329–346, Berlin, Heidelberg, 2008. Springer-Verlag.

[48] S.W. Huck. *Reading Statistics and Research*. Pearson, 2011.

[49] Yk Huhtala, Juha Krkkinen, Pasi Porkka, and Hannu Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(2):100–111, 1999.

[50] B. Iglewicz and D.C. Hoaglin. *How to detect and handle outliers*. ASQC basic references in quality control. ASQC Quality Press, 1993.

[51] Matthew A. Jaro. Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[52] R. A. Johnson and D. W. Wichern, editors. *Applied multivariate statistical analysis*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[53] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience, 9th edition, March 1990.

[54] William Kent. Limitations of record-based information models. *ACM Trans. Database Syst.*, 4(1):107–131, March 1979.

[55] Edwin M. Knorr and Raymond T. Ng. A unified notion of outliers: Properties and computation. In *In Proc. of the International Conference on Knowledge Discovery and Data Mining*, pages 219–222. AAAI Press, 1997.

[56] Flip Korn, Alexandros Labrinidis, Yannis Kotidis, and Christos Faloutsos. Ratio rules: A new paradigm for fast, quantifiable data mining. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 582–593, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[57] David Kravets. *TSA Admits Bungling of Airport Body-Scanner Radiation Tests*. WIRED, http://www.wired.com/threatlevel/2011/03/tsa-radiation-test-bungling¿, 15 Mar. 2011.

[58] Mong-Li Lee, Tok Wang Ling, and Wai Lup Low. Designing functional dependencies for xml. In *Proceedings of the 8th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '02, pages 124–141, London, UK, 2002. Springer-Verlag.

[59] M. Levene and A. Poulovanssilis. An object-oriented data model formalised through hypergraphs. *Data Knowl. Eng.*, 6:205–224, May 1991.

[60] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, CIKM '03, pages 556–559, New York, NY, USA, 2003. ACM.

[61] Shou-de Lin and Hans Chalupsky. Unsupervised link discovery in multi-relational data via rarity analysis. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 171–178, Washington, DC, USA, 2003. IEEE Computer Society.

[62] M. Loève. *Probability theory*. Number v. 1 in Graduate texts in mathematics. Springer-Verlag, 1978.

[63] Stéphane Lopes, Jean-Marc Petit, and Lotfi Lakhal. Efficient discovery of functional dependencies and armstrong relations. In *Proceedings of the 7th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '00, pages 350–364, London, UK, 2000. Springer-Verlag.

[64] Heikki Mannila and Kari-Jouko Räihä. Algorithms for inferring functional dependencies from relations. *Data Knowl. Eng.*, 12(1):83–99, 1994.

[65] Andrian Marcus and Jonathan I. Maletic. Utilizing association rules for the identification of errors in data. Technical report, 2000.

[66] Prabhaker Mateti and Narsingh Deo. On algorithms for enumerating all circuits of a graph. *SIAM J. Comput.*, pages 90–99, 1976.

[67] A. Maydanchik. *Data Quality Assessment*. Data quality for practitioners series. Technics Publications, 2007.

[68] Brian Mcbride. Jena: A semantic web toolkit. *IEEE Internet Computing*, 6:55–59, 2002.

[69] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 169–178, New York, NY, USA, 2000. ACM.

[70] S. Milgram. The small world problem. *Psychology Today*, May 1967.

[71] Michael Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *INTERNET MATHEMATICS*, 1:226–251.

[72] F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4):354–359, November 1983.

[73] Felix Naumann, Ulf Leser, and Johann-Christoph Freytag. Quality-driven integration of heterogeneous information systems. In *In VLDB Conference*, pages 447–458, 1999.

[74] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Phys. Rev. E*, 64, 2001.

[75] Eyal Oren, Sebastian Gerke, and Stefan Decker. Simple algorithms for predicate suggestions using similarity and co-occurrence. In *Proceedings of the 4th European conference on The Semantic Web: Research and Applications*, ESWC '07, pages 160–174, Berlin, Heidelberg, 2007. Springer-Verlag.

[76] Jason W. Osborne and Amy Overbay. The power of outliers (and why researchers should always check for them). *Practical Assessment, Research & Evaluation*, 9(6):1–12, 2004.

[77] Balaji Padmanabhan and Alexander Tuzhilin. A belief-driven method for discovering unexpected patterns. In *KDD*, pages 94–100, 1998.

[78] Marcus Paradies, Christian Lemke, Hasso Plattner, Wolfgang Lehner, Kai-Uwe Sattler, Alexander Zeier, and Jens Krueger. How to juggle columns: an entropy-based approach for table compression. In *Proceedings of the Fourteenth International Database Engineering and Applications Symposium*, IDEAS '10, pages 205–215, New York, USA, 2010. ACM.

[79] Leo L. Pipino, Yang W. Lee, and Richard Y. Wang. Data quality assessment. *Commun. ACM*, 45:211–218, April 2002.

[80] Cartic Ramakrishnan, William H. Milnor, Matthew Perry, and Amit P. Sheth. Discovering informative connection subgraphs in multi-relational graphs. *SIGKDD Explor. Newsl.*, 7(2):56–63, December 2005.

[81] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[82] Marta Sabou, Miriam Fernandez, and Enrico Motta. Evaluating semantic relations by exploring ontologies on the semantic web. pages 269–280, 2010.

[83] L. Sachs. *Applied statistics: a handbook of techniques*. Springer series in statistics. Springer-Verlag, 1984.

[84] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[85] Monica Scannapieco and Carlo Batini. Completeness in the relational model: a comprehensive framework. In InduShobha N. Chengalur-Smith, Louiqa Raschid, Jennifer Long, and Craig Seko, editors, *IQ*, pages 333–345. MIT, 2004.

[86] S. Schaffert, J. Eder, S. Gr"unwald, T. Kurz, M. Radulescu, R. Sint, and S. Stroka. KiWi–a platform for semantic social software. In *4th Workshop on Semantic Wikis, ESWC*, 2009.

[87] Abraham Silberschatz, Henry Korth, and S. Sudarshan. *Database Systems Concepts*. McGraw-Hill, Inc., New York, NY, USA, 5 edition, 2006.

[88] Ramakrishnan Srikant and Rakesh Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, SIGMOD '96, pages 1–12, New York, NY, USA, 1996. ACM.

[89] Markus Stocker, Andy Seaborne, Abraham Bernstein, Christoph Kiefer, and Dave Reynolds. Sparql basic graph pattern optimization using selectivity

estimation. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 595–604, New York, NY, USA, 2008. ACM.

[90] Nenad Stojanovic, Rudi Studer, and Ljiljana Stojanovic. An approach for the ranking of query results in the semantic web. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *The Semantic Web - ISWC 2003*, volume 2870 of *Lecture Notes in Computer Science*, pages 500–516. Springer Berlin / Heidelberg, 2003.

[91] Jiao Tao, Li Ding, and Deborah L. McGuinness. Instance data evaluation for semantic web-based knowledge management systems. In *42st Hawaii International International Conference on Systems Science (HICSS-42 2009), 5-8 January 2009, Waikoloa, Big Island, HI, USA*, pages 1–10. IEEE Computer Society, 2009.

[92] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal Of The Royal Statistical Society Series B*, 63(2):411–423, 2001.

[93] Johanna Volker, Denny Vr, York Sure, and Andreas Hotho. Learning disjointness. In *In: Proceedings of the 4th European Semantic Web Conference*, pages 175–189. Springer, 2007.

[94] Raphael Volz, Siegfried Handschuh, Steffen Staab, Ljiljana Stojanovic, and Nenad Stojanovic. Unveiling the hidden bride: deep annotation for mapping and migrating legacy data to the semantic web. *J. Web Sem.*, 1(2):187–206, 2004.

[95] Howard Wainer. Robust statistics: A survey and some prescriptions. *Journal of Educational Statistics*, 1(4):285–312, 1976.

[96] Richard Y. Wang and Diane M. Strong. Beyond accuracy: What data quality means to data consumers. *Research in computer science*, pages 5–33, 1996.

[97] E. Watkins and Denis Nicole. Named graphs as a mechanism for reasoning about provenance. In Xiaofang Zhou, Jianzhong Li, Heng T. Shen, Masaru Kitsuregawa, and Yanchun Zhang, editors, *Frontiers of WWW Research and Development - APWeb 2006*, volume 3841, chapter 99, pages 943–948. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[98] Grant E. Weddell. Reasoning about functional dependencies generalized for semantic data models. *ACM Trans. Database Syst.*, pages 32–64, 1992.

[99] Dawei Yin, Zhenzhen Xue, Liangjie Hong, and Brian D. Davison. A probabilistic model for personalized tag prediction. In *KDD*, pages 959–968, 2010.

[100] Yang Yu and Jeff Heflin. Detecting abnormal data for ontology based information integration. *International Workshop on Semantic Technologies for Information-Integrated Collaboration.*, pages 431–438, 2011.

[101] Yang Yu and Jeff Heflin. Extending functional dependency to detect abnormal data in rdf graphs. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist, editors, *International Semantic Web Conference (1)*, volume 7031 of *Lecture Notes in Computer Science*, pages 794–809. Springer, 2011.

[102] Yang Yu, Donald Hillman, Basuki Setio, and Jeff Heflin. A case study in integrating multiple e-commerce standards via semantic web technology. In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, pages 909–924, Berlin, Heidelberg, 2009. Springer-Verlag.

[103] Yang Yu, Yingjie Li, and Jeff Heflin. Detecting abnormal semantic web data using semantic dependency. *International Conference on Semantic Computing*, 0:154–157, 2011.

[104] Yang Yu, Xingjian Zhang, and Jeff Heflin. Learning to detect abnormal semantic web data. In Mark A. Musen and Óscar Corcho, editors, *K-CAP*, pages 177–178. ACM, 2011.

[105] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: A new data clustering algorithm and its applications. *Data Min. Knowl. Discov.*, 1:141–182, January 1997.

Vita

Yang Yu completed his Ph.D. in Computer Science from Lehigh University, PA, USA in May, 2012. He holds a Bachelor of Engineering in Computer System from Southeast University, Nanjing, China, and a Master of Engineering in Computer Science from Nanjing University of Science and Technology, Nanjing, China. Yang's primary research interests are the Semantic Web, Data Mining, Machine Learning, Ontology Evaluation, Knowledge Extraction, Information Integration, etc. He has co-authored several papers in these domains, including the following:

Y. Yu, J. Heflin, Extending Functional Dependency to Detect Abnormal Data in RDF Graphs, the 10th International Semantic Web Conference (ISWC2011), pp794-809, Bonn, Germany, 2011.

Y. Yu, Y. Li, J. Heflin, Detecting Abnormal Semantic Web Data Using Semantic Dependency, the Fifth IEEE International Conference on Semantic Computing (ICSC 2011), pp154-157, Palo Alto, CA, USA, 2011.

Y. Yu, J. Heflin, Learning to Detect Abnormal Semantic Web Data, the Sixth International Conference on Knowledge Capture (K-CAP 2011), pp177-178, Banff, Canada, 2011.

Y. Yu, J. Heflin, Detecting Abnormal Data for Ontology Based Information Integration, International Workshop on Semantic Technologies for Information-Integrated Collaboration (STIIC 2011), CTS 2011, pp431-438, Philadelphia, PA, USA, 2011.

Y. Li, Y. Yu and J. Heflin, A Multi-ontology Synthetic Benchmark for the Semantic Web, the 1st International Workshop on Evaluation of Semantic Technologies (IWEST2010), ISWC2010, Shanghai, China, 2010.

Y. Yu, D. Hillman, B. Setio and J. Heflin, A Case Study in Integrating Multiple E-commerce Standards via Semantic Web Technology, the 8th International Semantic Web Conference (ISWC2009), pp 909-924, Washington D.C., USA, 2009.