

Thesis for the Degree of Doctor of Philosophy

# **Hybrid Similarity Measure for XML Data Integration and Transformation**

**Pham Thi Thu Thuy**

Department of Computer Engineering

Graduate School

Kyung Hee University

Seoul, Korea

August, 2012

# **Hybrid Similarity Measure for XML Data Integration and Transformation**

**Pham Thi Thu Thuy**

Department of Computer Engineering

Graduate School

Kyung Hee University

Seoul, Korea

August, 2012

# **Hybrid Similarity Measure for XML Data Integration and Transformation**

by

**Pham Thi Thu Thuy**

Advised by

**Professor Young-Koo Lee**

**Professor Sungyoung Lee**

Submitted to the Department of Computer Engineering  
and the Faculty of the Graduate School of  
Kyung Hee University in partial fulfillment  
of the requirements for the degree of  
Doctor of Philosophy

## **Dissertation Committee:**

Professor Byeong-soo Jeong, Ph.D. ....

Professor Brian J. d' Auriol, Ph.D. ....

Professor Jin-Ho Kim, Ph.D. ....

Professor Donghai Guan, Ph.D. ....

Professor Young-Koo Lee, Ph.D. ....

# Hybrid Similarity Measure for XML Data Integration and Transformation

by

**Pham Thi Thu Thuy**

Advised by

**Professor Young-Koo Lee**

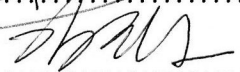
**Professor Sungyoung Lee**

Submitted to the Department of Computer Engineering  
and the Faculty of the Graduate School of  
Kyung Hee University in partial fulfillment  
of the requirements for the degree of  
Doctor of Philosophy


## Dissertation Committee:

Professor Byeong-soo Jeong, Ph.D. .... 

Professor Brian J. d'Auriol, Ph.D. .... 

Professor Jin-Ho Kim, Ph.D. .... 

Professor Donghai Guan, Ph.D. .... 

Professor Young-Koo Lee, Ph.D. .... 

# Hybrid Similarity Measure for XML Data Integration and Transformation

by

Pham Thi Thu Thuy

Submitted to the Department of Computer Engineering  
on July 8, 2012, in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy

## Abstract

XML (eXtensible Markup Language) has been widely used as a standard for sharing data between web-based applications. In order to share XML data with another XML application system, it is needed to integrate various XML data sources into a coherent XML data set. Moreover, to share XML data with semantic supporting system, such as Web Ontology Language (OWL), XML data also need to be transformed into OWL ontology. However, since the heterogeneous of XML data, the same information can be published in many different ways in terms of tag names and structures or the same tag names can represent different contents, the sharing of XML data is not yet fully automatic. This heterogeneity of XML data has led to research in measuring the similarity of elements between XML schemas or element similarity within a schema. Therefore, to perform the integrating and transforming tasks, the similarity measure of XML schema play a crucial role due to the heterogeneous of XML data sources.

In this thesis, we deal with the problem of data transformation and integration for XML data sources. This data format presents a lot of challenges that need XML-specific solutions: an XML schema is not required for an XML document, and if XML schema exists, it may be expressed in a number of different XML schema types such as XML Schema (XSD) or Document Type Definition (DTD) ; also resolving the heterogeneity in the schema is not straightforward method due to the hierarchical nature of XML data.

We propose a hybrid similarity measure based approach, that handles the distinct problems of syntactic, semantic, and schematic heterogeneity of XML data. Our similarity measure addresses both structural and semantic components and can be applied for both schema types of XML . Due to the different targets between integration and transformation of XML data, we propose two types of similarity measures, which are similarity of elements between two schemas for data integration and similarity of elements within a schema for data transformation. In particular, we can divide the thesis into two main parts, both related to enhance the sharing of XML data.

The first part focuses on the similarity measure between schemas for data integration. We propose the novel similarity measure that concurrently considers both structural and semantic in-

formation of two specific XML Schemas. Specifically, we introduce new metrics to compute the data type and cardinality constraint similarities which improve the quality of the current semantic assessments. Based on the similarity between element pairs, we put forward an algorithm to calculate the similarity between two XML Schema trees. Based on the similarity measure, we propose an integration method to merge two or more disparate XML data sources into a single coherent data set to support the information needs of the target business or enterprise. Experimental results lead to the conclusion that our methodology provides better similarity values than the others with regard to the accuracy of semantics and structure similarities.

The second part of the work is related to the similarity measure of duplicate within a schema and the transformation of XML Schema into OWL (Web Ontology Language). This part is also divided into two different sub-parts. The first one is focused on the problem of duplicate elements in XML Schema. Recent studies on transforming XML Schema into OWL have shown that the associated duplicates problem can be solved by creating a unique identifier for each element. However, this solution considers duplicate elements to be different nodes, whereas most duplicates represent the same information. We present a novel method to measure the semantic similarity between duplicate elements within an XML Schema. Semantic similarity is the combination of the declaration and context features, which capture all the descriptions and relationships of the duplicate elements. Based on the similarity values, we classify the duplicates into two groups: similar and non-similar, and then propose the suitable strategy to transform these duplicates into appropriate OWL concepts.

In the second sub-part, we present a mechanism to ease the interpretation and automate the semantic transformation of specific XML data into the OWL ontology (S-Trans), which allows an easier and better semantic communication among information systems. On the basis of the XML schemas (XSD or DTD), we extract the document structure and add more descriptions for XML elements. Experimental results show that the proposed method reliably predicts semantic similarity of duplicates and produces a better-quality of OWL ontology.

Thesis Supervisor: Young-Koo Lee  
Title: Professor

Thesis Co-Supervisor: Sungyoung Lee  
Title: Professor

## Acknowledgments

There are countless people who have supported, directed, assisted and encouraged me in completing this PhD, and that I would like to thank.

First of all, I would like to express my deepest gratitude to my supervisor, Professor Young-Koo Lee, for supervising my work and for always having the right suggestions during any discussion we had. He has not only led me to the research area of the semantic similarity measure for data integration but also offered me lots of insightful suggestions based on which I have developed and completed my dissertation.

I would like to thank Kyung Hee University and IITA scholarship as a whole to have given me the opportunity - and have provided the funds - to carry out this PhD. I also grateful to my co-supervisor, Professor Sungyoung Lee, who guided me toward the proper direction with his inquisitive questions and helpful comments. I also would like to thanks Professor Brian J.d' Auriol's for his advices on my presentation and visualization skill have been going with me not only in the past but also at the present and absolutely in my future research career.

I would like to thank number of Professors in Computer Engineering Department for their excellent lectures, Professor Tae-Choong Chung, Professor Ok-Sam Chae, Professor Byeong-Soo Jeong, Professor Choong-Seon Hong, and Professor Eui-Nam Huh. Their wisdom greatly contributed to consolidating and widening my knowledge on computer engineering, which is also the very important background for my dissertation.

My thanks also go to many of my colleagues who help and encourage me during my stay in Korea. Especially, Prof. Dr. Donghai Guan, Dr. Phan Tran Ho Truc, Dr. Dang Viet Hung, senior Le Tuan Anh, senior Nguyen Hoang Viet, senior Vo Thi Luu Phuong, the couple Nguyen Van Mui + Tran Thi Kim Loc, Korean friends Yongkoo Han, Jinseung Kim, Kisung Park, and lab-mates La The Vinh, Dinh Dong Luong, Pham The Anh, Iram Fatima, and many others, who have shared

their knowledge and technical expertise with me. Without their suggestions, my life and my research would have been much harder.

I also would like to express my deeply thanks to the dissertation committee members whose helpful comments have helped me to improve and complete this dissertation.

Last but not least my thanks go to my family of course, to whom I dedicate this achievement: my parents and my parents in law, who, with love and comprehension, has always pushed me to pursue a PhD. I would like to send my love to my sweet husband and my two lovely daughters, Bin and Su, who always stay beside me during a somewhat stressful time.

Thanks to you all. ♡ .



# Contents

<b>Table of Contents</b>	<b>v</b>
<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 XML data sharing scenario . . . . .	4
1.3 Motivation and contributions . . . . .	6
1.4 Thesis Organization . . . . .	9
<b>2 Background and Related Work</b>	<b>11</b>
2.1 Background on XML Data and OWL Ontology . . . . .	11
2.1.1 XML data . . . . .	11
2.1.2 Ontology . . . . .	12
2.1.3 OWL fundamental constructs . . . . .	16
2.1.4 Term definitions . . . . .	19
2.2 Related Work . . . . .	20
2.2.1 Similarity between documents and XML integration . . . . .	20
2.2.2 Duplicate similarity and XML schema transformation . . . . .	28

<b>3</b>	<b>ESim: Element Similarity measure for XML integration</b>	<b>32</b>
3.1	Similarity Measure Framework . . . . .	32
3.2	Semantic Similarity Measurement (SeSim) . . . . .	35
3.2.1	Name similarity (NSim) . . . . .	38
3.2.2	Data type similarity . . . . .	40
3.2.3	Constraint similarity . . . . .	43
3.3	Structural Similarity Measurement (StSim) . . . . .	46
3.3.1	Ancestor similarity . . . . .	47
3.3.2	Sibling similarity . . . . .	48
3.3.3	Children similarity . . . . .	49
3.4	Similarity between Two Schema Trees . . . . .	50
3.5	XML Schema Integration . . . . .	52
<b>4</b>	<b>S-Trans: Duplicate Similarity Measure for XML2OWL</b>	<b>55</b>
4.1	General modules of XML2OWL Transformation . . . . .	55
4.2	Semantic Similarity of Duplicate Elements . . . . .	57
4.2.1	Motivating example . . . . .	57
4.2.2	Ancestor similarity (ASim) . . . . .	61
4.2.3	Sibling similarity (SbSim) . . . . .	62
4.2.4	Children similarity (ChSim) . . . . .	63
4.3	Transforming DTD/XSD into the OWL Ontology . . . . .	65
<b>5</b>	<b>Experimental Results</b>	<b>71</b>
5.1	Experiments on XML Schemas Similarity Measure . . . . .	72
5.1.1	Determining of parameter values . . . . .	72
5.1.2	Results based on real-world XSDs . . . . .	76
5.2	Experiments on Duplicate Similarity in XML Transformation . . . . .	84
5.2.1	Experimental setup . . . . .	84
5.2.2	Results . . . . .	85

5.3	Experimental Summarization . . . . .	91
<b>6</b>	<b>Conclusion and Future Researches</b>	<b>93</b>
6.1	Conclusion . . . . .	93
6.1.1	Thesis summary . . . . .	93
6.1.2	Contributions . . . . .	95
6.2	Future Researches . . . . .	97
	<b>Appendix A: ESim - Evaluation Results</b>	<b>100</b>
	<b>Appendix B: Sample of XML Schema for Transformation</b>	<b>103</b>
	<b>Appendix C: OWL Ontology Result</b>	<b>108</b>
	<b>References</b>	<b>118</b>

# List of Figures

1.1	Semantic Web stack architecture . . . . .	2
1.2	Different solutions to integration and transformation XML data . . . . .	3
1.3	Thesis organization. . . . .	10
2.1	Example of a XML document and its respective DTD . . . . .	13
2.2	Example of the respective XSD of document in Figure 2.1 . . . . .	14
2.3	OWL root classes . . . . .	16
2.4	OWL subclass definition . . . . .	17
2.5	OWL class individual . . . . .	18
2.6	OWL Datatype property definition . . . . .	18
2.7	OWL Class instance with datatype property . . . . .	19
3.1	General framework of similarity measure method . . . . .	33
3.2	Tree representation for Schema Patient_A . . . . .	34
3.3	Tree representation for Schema Patient_B . . . . .	35
3.4	Expressions for Schema Patient_A . . . . .	37
3.5	A fragment of WordNet . . . . .	39
3.6	The structure similarity algorithm . . . . .	51
3.7	XML Schema integration framework architecture . . . . .	53
4.1	General syntactic to semantic architecture . . . . .	56
4.2	Example of a DTD document, prescription.dtd . . . . .	58

4.3	Example of a DTD and a part of its corresponding XSD document . . . . .	59
4.4	The corresponding tree of XML schema (XSD/DTD) in Figure 4.2 . . . . .	60
4.5	The ancestor similarities at different ancestor levels with five candidate values . .	62
4.6	The ancestor similarity algorithm . . . . .	63
4.7	Transforming framework from XML into OWL . . . . .	66
4.8	The transforming correspondences between DTD/XSD and OWL . . . . .	68
4.9	OWL results of duplicates which are highly similar . . . . .	70
4.10	OWL results of duplicates which are less similar . . . . .	70
5.1	Tree representation for Schema Patient_C . . . . .	74
5.2	Tree representation for Schema Patient_D . . . . .	74
5.3	Determining weights of ESim function . . . . .	75
5.4	Determining weights of SeSim function . . . . .	75
5.5	Matching comparisons of ESim to COMA, XMLSim, and XClust . . . . .	80
5.6	Quality of name measure . . . . .	81
5.7	Quality of data type measure . . . . .	82
5.8	Quality of cardinality constraint measure . . . . .	82
5.9	Quality of structure measure . . . . .	83
5.10	F_measure comparison . . . . .	83
5.11	The error rate of classification at different thresholds . . . . .	86
5.12	Evaluation results, <i>drug_medicament</i> schema . . . . .	87
5.13	Evaluation results, <i>patient_admission</i> schema . . . . .	88
5.14	Evaluation results, <i>healthcaremetadata</i> schema . . . . .	88
5.15	Evaluation results, <i>pathology.report</i> schema . . . . .	89
5.16	Quality of <i>S-Trans</i> , <i>PrSim</i> , <i>ChSim</i> , and <i>CaSim</i> . . . . .	90
A.1	Evaluation results of matching system for schemas in Table 5.2 . . . . .	101

# List of Tables

3.1	Data type compatibility table . . . . .	44
3.2	Cardinality constraint similarity table . . . . .	45
5.1	The similarity of synthetic XSDs . . . . .	73
5.2	The characteristics of the tested schemas . . . . .	76
5.3	Element similarity result of the two schemas (Patient_A and Patient_B) . . . . .	78
5.4	The characteristics of the tested schemas . . . . .	85

# Chapter 1

## Introduction

### 1.1 Introduction

Recently, many web-page applications and services publish their data using XML, the standard for sharing data, since the use of XML as a common data representation format helps sharing XML data with other applications and services. Usually, to improve the sharing of XML data with the same XML application system, all XML data sources are integrated into a coherent data set to support the information needs of the target applications. Moreover, to enhance the sharing of XML data with the semantic supporting system using OWL, XML data are transformed into the target OWL ontology. However, since the heterogeneous of XML data in which the same information can be published using XML in many different ways in terms of tag names and structures or the same tag names can represent different contents, the exchange of XML data is not fully automatic. To solve the heterogeneity problem of XML data, many researches have been proposed similarity measure methods to compute the similarity of heterogeneous XML data before integrating or transforming them. The algorithms that automate these similarity computations help to reduce time and effort spent on creating and maintaining data sharing in many applications [90] such as in e-Business [12], [100], [93], e-Government [70], [73], [48], [25], [33], e-Learning [18], [11], [98], and e-Health [95], [88], [68].

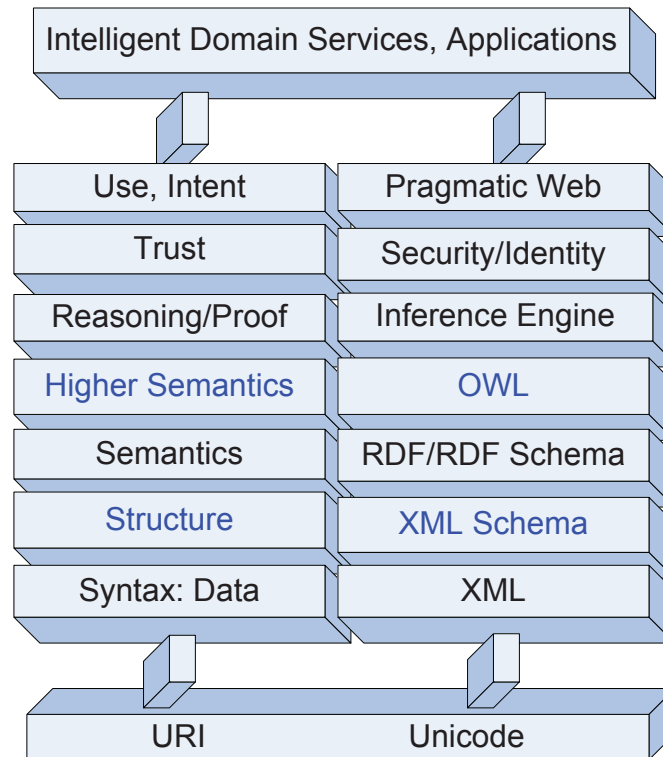


Figure 1.1: Semantic Web stack architecture

To illustrate the important of XML data integration, let us take one integration example in e-Health system. In the e-Health system, there are various of XML healthcare data. These data are the collection of healthcare data from the large number of environmental and patient sensors, and actuators to monitor and improve patient's physical and mental conditions [86]. Nowadays, the XML healthcare data are increasing, so the healthcare providers need to integrate these data in order to keep them as the electronic health record (EHR) [32]. Therefore, the integration of XML healthcare data plays an important role in enhancing the quality of the patient care and the information exchange among the medical systems.

In general, although heterogeneous XML sources may have similar content, they may be de-



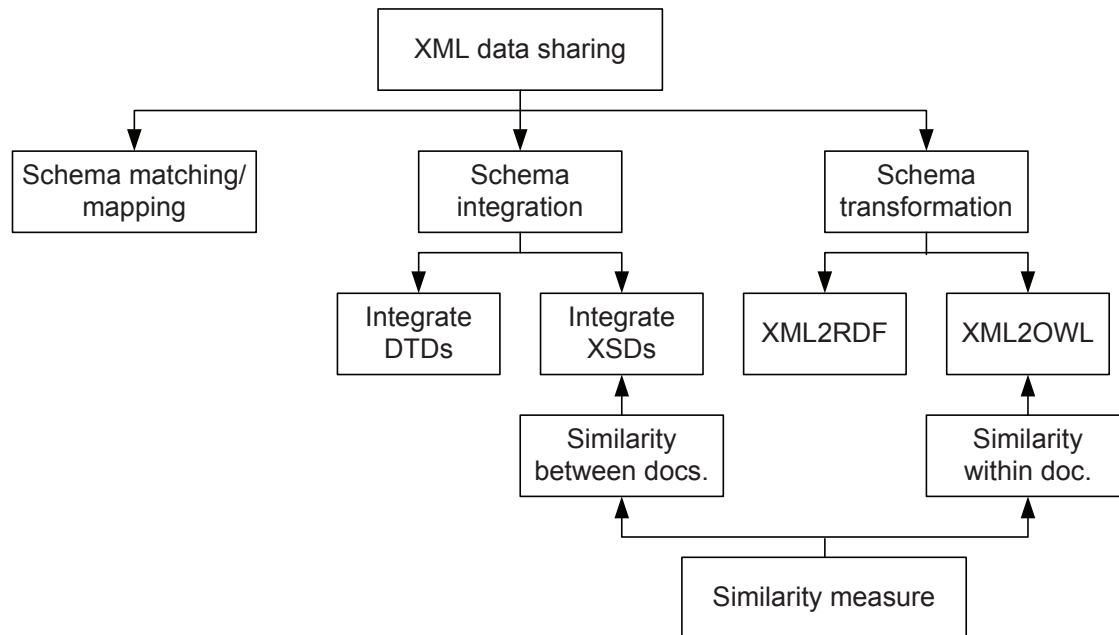


Figure 1.2: Different solutions to integration and transformation XML data

scribed using different tag names and structures. Integration of similar XML documents from different data sources benefits applications which use the same XML language, giving them access to more complete and useful information and query systems to retrieve information from a single integrated source instead of various sources.

On the other hand, recently, the Semantic Web has been developed and widely used by many semantic applications. This development leads to the need for sharing the existing XML data with semantic applications. However, XML is disadvantage when it comes to the semantic interoperability because it focuses primarily on the syntactics, with no way to describe the semantics of the data [34]. This lack of semantic description leads to the problems when semantic agents seek to understand and reason about these XML data. Therefore, to enable the sharing of XML data with semantic supporting systems, it is needed to map or transform XML data into a semantic

supporting language. In this thesis, we choose OWL as the target source for the transformation, since OWL is described as higher semantic language in the Semantic Web stack architecture [36]. Moreover, since the heterogeneous of XML data where duplicates may represent different or same information, to improve the semantics of the transformation, we propose a pre-step to compute the semantic similarity of XML elements, specifically the duplicate elements, before the XML transforming process.

In general, this thesis tackles the problem of sharing XML data between the same XML applications and between XML application and the semantic supporting application. In particular, we have developed an approach to the integration and transformation of heterogeneous XML data sources. Our approach is based on the similarity measure method, meaning that the output is a set of similarity scores of elements between XML schema documents, in an XML data integration scenario, or a set of similarity scores of duplicate elements within XML schema, in an XML data transformation scenario. The overview of different solutions to enhance the data sharing and our focused research is illustrated in Figure 1.2.

The rest of this chapter is constructed as follows. Section 1.2 introduces the different scenarios in the broad area of data sharing. Section 1.3 presents the motivation and contributions of the work described in this thesis. Section 1.4 gives an overview of the thesis organization.

## 1.2 XML data sharing scenario

The sharing of XML data across applications and services may involve several scenarios, including: *XML schema integration* and *XML schema transformation*. However, all scenarios share the same process of similarity measure, particularly, *similarity between documents* for the integration scenario and *similarity within a document* for the transformation scenario. We introduce below

the major scenarios and processes in schema integration and transformation.

*XML schema integration* is an XML data sharing scenario in which XML data from multiple data sources are combined in order to give users a single integrated source. This task may retain all of the original logical structures and tag names of the XML schema sources (XSD or DTD), since it generates a union or global XML schema which combines the data sources in more complex ways.

*XML schema transformation* is an XML data sharing scenario in which one needs to define rules for transforming a source XML schema  $S_1$  and its associated XML instances  $DS_1$  to a structure of the target schema  $S_2$  which is defined in a different modeling language as  $S_1$ , for the purposes of query processing or materialization of  $S_2$ , using the data  $DS_1$ . *XML data exchange* is a stricter form of XML data transformation, which also respects the constraints defined within the target XML schema, and not just its structure.

*Element similarity between XML schemas* is the automatic or semi-automatic process of determining the similarity scores between elements of an XML schema  $S_1$  and those of another XML schema  $S_2$ . The next step of this process is the classification process in which highly similarity element pairs are combined into an integrated source. The process of choosing a classification value is discussed in the experiment section.

*Similarity of elements within an XML schema* is the automatic or semi-automatic process of determining the similarity scores between elements within a schema  $S_1$ . The similarity results can then be used to transform data from the data source of  $S_1$  into  $S_2$ . In this thesis, we compute the similarity value of duplicate elements in an XML Schema and then classify them into the similar or non-similar group for the transformation.

### 1.3 Motivation and contributions

From the above overview, a number of research questions arise regarding XML data sharing, which form the motivation for our research:

- How to improve the data sharing between applications using the same XML system or sharing XML system with higher semantic supporting language, OWL?
- Different XML data sources may be associated with different XML schema types, or may not have a same schema type at all. Can we encompass all types of XML data sources with a data transformation or an integration approach?
- How to solve the heterogeneous problem of XML data during the integration or transformation XML data?
- Which aspects of XML data transformation and integration can be automated? Are they clearly distinguishable from the manual aspects? Can we minimize the manual aspects?
- XML data sources may be structurally incompatible, which may lead to loss of information when transforming or integrating them. How to solve this problem automatically?
- Have existing approaches performed the integration or transformation of XML data? If so, do they have any problem needed to resolve?

With these research questions as a starting point, this thesis proposes a similarity measure based approach for the integration and transformation of heterogeneous XML data sources and makes the following contributions:

1. We propose the integration method-based similarity measure to improve the data sharing between the same XML applications. For sharing data with higher semantic application, we

propose the transformation of XML into OWL ontology method with consider the duplicate similarity in XML schema.

2. Our approach can be applied on any type of XML data sources, regardless of the schema type used, XSD or DTD.
3. We propose a hybrid similarity measure to compute both semantic and structural similarities of XML elements.
4. We automate the similarity measure process for data integration and transformation by providing the metrics to compute all similarity factors. There is no similarity value given by users. Our propose metrics generates more precise similarity values than those by manual. Moreover, we minimize the manual aspect by proposing the method to determine the weighted values to balance the role of the similarity factors.
5. To solve the loss of information problem, in the integration process, our integrator take a union of all elements in XML schemas instead of retaining only common elements. In the transformation process, we follow the structural descriptions of XML schemas to transform all elements and their relationships with other elements into appropriate OWL concepts.
6. There are several approaches proposed to integrate and transform XML data. However, our methods are overcome the existing work because of some reasons. For the integration approaches, in most of related approaches, the data type, cardinality constraint, and weight parameters values are manually given whereas we provide novel metrics to determine those values. In the transformation approaches, most existing methods solve the duplicate problem of XML data by simply giving each XML element a unique identifier, which may cause the redundancy data when duplicates represent the same information. We resolve this duplicate problem by proposing the duplicate similarity measuring and giving an appropriate strategy to transform them.

With respect to existing approaches to XML schema transformation and integration, our approach makes a number of contributions:

1. We propose a new metric to measure the data type similarity between two attribute types whereas data type similarity value is given manually in related work.
2. We present the novel metric to measure the similarity of the cardinality constraints which are also manually given by user.
3. In order to avoid the case that two nodes have the same structure but difference in their names, we compute the structural similarity of two concepts by relying on the semantic similarity and each pair of their neighborhood elements.
4. We present an algorithm to calculate the similarity between two schema trees based on the similarity values of the element pairs.
5. We propose a method to determine the weight parameters which are used to balance the role of the similarity measuring factors.
6. We discovers the semantic problem during transformation of duplicate elements in an XML schema into ontology.
7. We proposes method to measure the semantic similarity between repeated elements, which considers not only the relationship similarity, but also the inside descriptions of each duplicate node.
8. We propose a method to formally determine the duplicate classifying value.
9. It proposes the strategy to transform XML schema and their duplicates into ontology.
10. Finally, our approach addresses the problem of human intervention during the integration and data redundancy in transformation of XML data. Experimental results reveal that our method overcomes the related work in terms of semantics and accuracy.

## 1.4 Thesis Organization

This section describes the the road map for the entire thesis. We provide the thesis organization in Figure 1.3. A brief summary of each chapter is shown below.

- **Chapter 1 Introduction.** This chapter briefly introduces the population of XML data and an example of XML in e-Health system. The challenges and disadvantages of XML's flexibility in creating new document and lack of semantics support of XML are clearly addressed. After that the dissertation focuses and contributions are also made clear.
- **Chapter 2 Background and Related Work.** This chapter presents to sections. First, we review background knowledge on XML data and OWL ontology. Second, we give a comprehensive survey of the existing work especially work that relates to two problems: measuring the similarity between XML Schema documents and transforming XML into OWL ontology. The state of the art and limitations of existing work are clearly addressed.
- **Chapter 3 Semantic and Structural Similarity between XML Schemas.** The proposed solution for the semantic and structural measuring problem is described in detail in this chapter.
- **Chapter 4 Duplicate and Transforming XML schemas into OWL ontology.** This chapter describes all the details of the semantic similarity measuring for duplicate elements in XML schemas and proposes solution for each similarity level and transforms all XML schemas elements into OWL ontology.
- **Chapter 5 Experimental results and discussions.** Comprehensive experiments are conducted, the results are analyzed to enlighten the advantages of the proposed algorithms.
- **Chapter 6 Conclusion and future work.** In this chapter, a conclusion is given. Besides, some limitations of the work are also pointed out with potential solutions, which may need further research effort to be completed.

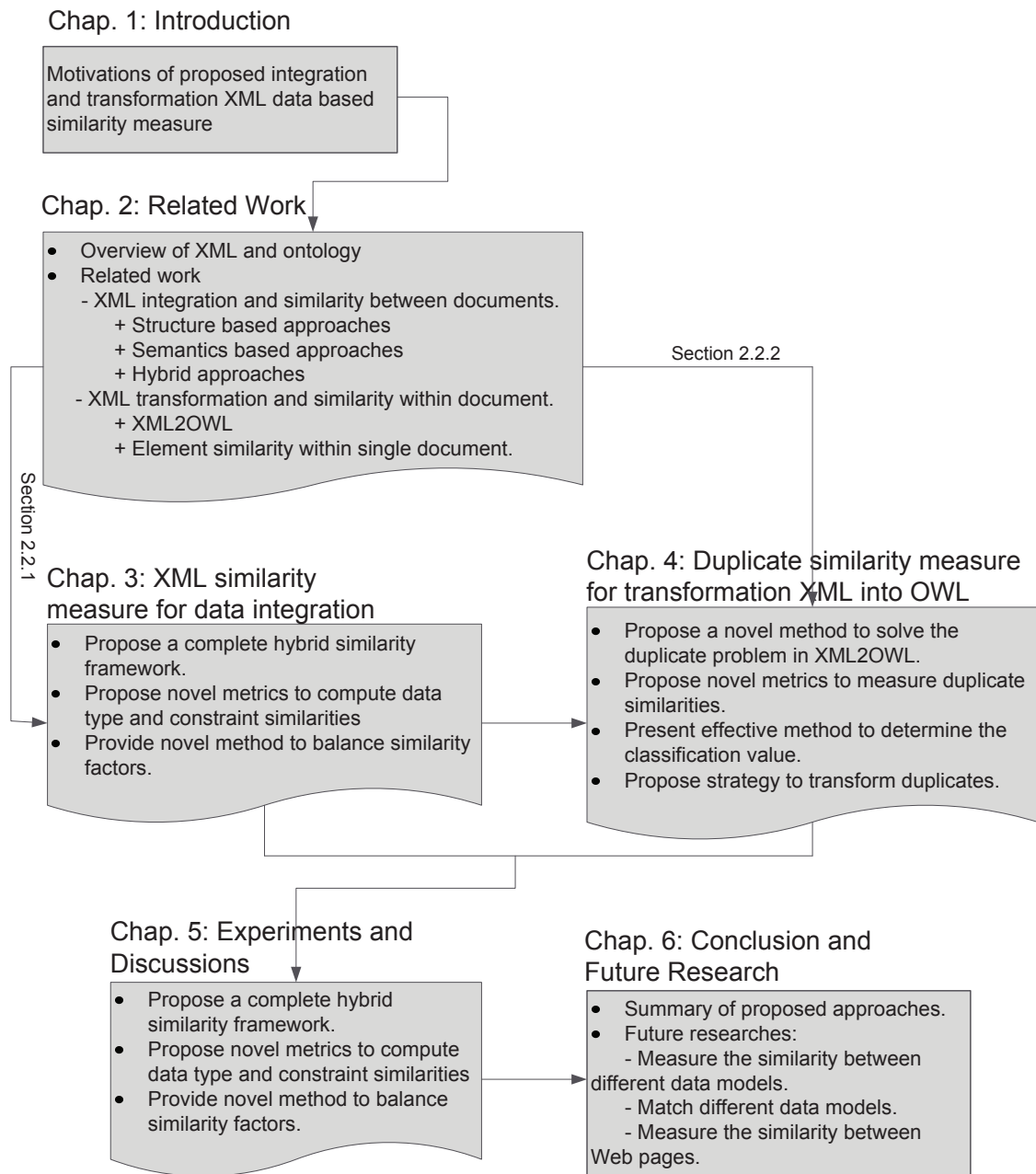


Figure 1.3: Thesis organization.



## **Chapter 2**

# **Background and Related Work**

Since XML data and ontology are two main objects in this dissertation, in this chapter we give a brief introduction to their characteristics and technologies. After that, we discuss the related researches to our work.

## **2.1 Background on XML Data and OWL Ontology**

### **2.1.1 XML data**

XML (eXtensible Markup Language) is a flexible representation language. There are two varieties of XML data: XML documents and XML schemas. An XML schema provides the data definitions and structure of the XML document [65]. While XML documents are the instances of an XML schema which gives a snapshot of what the document may contain. A schema includes what elements are allowed or are not allowed; what attributes for any elements may be and the number of occurrences of XML elements; etc. A schema for a document may be included as both internally (located within the schema document) and externally (independently located outside XML schema file).

There are several XML schema languages, but only two are commonly used. They are DTD (Document Type Definition) and XML Schema or XML Schema Definition (XSD), both of which allow the construction of XML documents to be described and their contents to be constrained [79]. A DTD specifies the structure of an XML element by specifying the names of its sub-elements and attributes. Subelement structure is specified using some operators, such as \* (zero or more elements), + (one or more elements), ? (optional), and | (or), as well as with properties type (PCDATA, ID, IDREF, ENUMERATION).

The DTD language is disadvantaged in compare with an XSD language since it only supports a limited set of data types, has loose structure constraints, uses different language with XML, etc. To overcome the above limitations of DTD, the XSD language provides the novel features, such as simple and complex types, rich data type sets, occurrence constraints and especially using the same language with XML. An XML Schema is usually comprised of a set of schema components, such as the data type definitions and cardinality constraint declarations, etc. They can be used to evaluate the validity of the well-formed element information items. It is believed that XSD will soon replace DTD due to its flexibility [41]. Throughout this thesis, we use the term *XML schema* to express both the DTD and XSD, while XML Schema represents the XSD.

Figure 2.1 illustrates a simple example of a XML document and its corresponding DTD. Figure 2.2 shows a respective XML Schema.

### 2.1.2 Ontology

In computer science, an *ontology* is "an explicit specification of a conceptualization" [31], *i.e.* an ontology is a model that describes the concepts of a problem domain, as well as the association between those concepts. An ontology can be used as an interface to one or more data sources which means that it can be used as a schema, or it can be used to reason about the problem domain.

<pre> &lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;Companies&gt;   &lt;Company&gt;     &lt;Symbol&gt; Eagle.img &lt;/Title&gt;     &lt;Name&gt; EagleFarm &lt;/Name&gt;     &lt;Industry&gt; Dairy &lt;/Industry&gt;     &lt;Profile&gt;       &lt;MarketCap&gt; 1000 &lt;/ MarketCap &gt;       &lt;EmployeeNo&gt; 20 &lt;/ EmployeeNo &gt;       &lt;Address&gt;         &lt;State&gt; QLD &lt;/State&gt;       &lt;/Address&gt;       &lt;Description&gt; gdsfkl &lt;/Description&gt;     &lt;/Profile&gt;   &lt;/Company&gt;   &lt;!-- Some more instances --&gt;   .... &lt;/Companies&gt; </pre>	<pre> &lt;!DOCTYPE Companies [   &lt;!ELEMENT Companies (Company+)&gt;   &lt;!ELEMENT Company (Symbol, Name,     Sector?, Industry, (Profile))&gt;   &lt;!ELEMENT Profile (MarketCap,     EmployeeNo, (Address),     Description)&gt;   &lt;!ELEMENT Address (State,City?)&gt;   &lt;!ELEMENT Symbol(#PCDATA)&gt;   &lt;!ELEMENT Name (#PCDATA)&gt;   &lt;!ELEMENT Sector (#PCDATA)&gt;   &lt;!ELEMENT Industry (#PCDATA)&gt;   &lt;!ELEMENT MarketCap (#PCDATA)&gt;   &lt;!ELEMENT EmployeeNo (#PCDATA)&gt;   &lt;!ELEMENT State (#PCDATA)&gt;   &lt;!ELEMENT City (#PCDATA)&gt; ]&gt; </pre>
--	--

Figure 2.1: Example of a XML document and its respective DTD

*RDF* (Resource Description Framework) [64] is a family of W3C specifications which is used primarily for specifying the information about a problem domain. *RDF* has the triple form of subject-predicate-object. Therefore, a set of *RDF* statements generates a labeled, directed graph. *RDF Schema* is one of the W3C *RDF* specifications. *RDF Schema* allows the definition of *RDF* vocabularies. Note that *RDF* can also be used as the data format for the exchange and integration of data from different information systems.

*OWL* (Web Ontology Language) [37], like *RDF Schema*, is used to define ontologies. *OWL* is also a Semantic Web language designed to represent more rich and complex knowledge about things, groups of things, and relations between things than *RDF*. *OWL* is a logic-based language so knowledge expressed in *OWL* can be reasoned with by computer programs either to verify the consistency of that knowledge or to understand about the expressed knowledge. The *OWL* doc-

---

```
1 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
2   <xsd:element name="Companies">
3     <xsd:complexType>
4       <xsd:sequence>
5         <xsd:element name="Company" maxOccurs="unbounded">
6           <xsd:complexType>
7             <xsd:sequence>
8               <xsd:element name="Symbol" type="xsd:string"/>
9               <xsd:element name="Name" type="xsd:string"/>
10              <xsd:element name="Sector" type="xsd:string"/>
11              <xsd:element name="Industry" type="xsd:string"/>
12              <xsd:element name="Profile">
13                <xsd:complexType>
14                  <xsd:sequence>
15                    <xsd:element name="MarketCap" type="xsd:string"/>
16                    <xsd:element name="EmployeeNumber" type="xsd:unsignedInt"/>
17                    <xsd:element name="Address">
18                      <xsd:complexType>
19                        <xsd:sequence>
20                          <xsd:element name="State" type="xsd:string"/>
21                          <xsd:element name="City" type="xsd:string"/>
22                        </xsd:sequence>
23                      </xsd:complexType>
24                    </xsd:element>
25                    <xsd:element name="Description" type="xsd:string"/>
26                  </xsd:sequence>
27                </xsd:complexType>
28              </xsd:element>
29            </xsd:sequence>
30          </xsd:complexType>
31        </xsd:element>
32      </xsd:sequence>
33    </xsd:complexType>
34  </element>
35 </xsd:schema>
```

---

Figure 2.2: Example of the respective XSD of document in Figure 2.1

uments, known as ontologies, can be distributed in the World Wide Web and may refer to or be referred from other OWL ontologies.

The OWL language has three increasingly expressive sublanguages as following:

**OWL Lite** [59], [1] supports those users primarily needing a classification hierarchy and simple constraint features. For example, the cardinality constraints in OWL Lite only allows cardinality values of 0 or 1. Thus, OWL Lite provides a quick migration path for thesauri and other taxonomies.

**OWL DL** [71], [60] provides those users who want the maximum expressiveness without losing computational completeness and all computations, which will finish in finite time, of the reasoning systems. OWL DL includes all the OWL language constructs with restrictions such as type separation (for instances, a class cannot also be an individual or property, a property cannot also be an individual or class). OWL DL is so named due to its correspondence with description logics, a field of research that has studied a particular decidable fragment of first order logic. OWL DL was designed to support the existing Description Logic business segment and has desirable computational properties for the reasoning systems.

**OWL Full** [44], [15] is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals. Another significant difference from OWL DL is that an OWL full data type property may be inverse functional. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support every feature of OWL Full.

Each of these sublanguages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded. The following set of relations hold.

- Every legal OWL Lite ontology is a legal OWL DL ontology.

---

```

1    <owl:Class rdf:ID="RedWine"/>
2    <owl:Class rdf:ID="Winery"/>

```

---

Figure 2.3: OWL root classes

- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

### 2.1.3 OWL fundamental constructs

In this section, we will present the fundamental elements of OWL, which include the classes, properties and individuals. Every OWL construct is uniquely defined by an *rdf:ID*. The **OWL classes** describe sets of individuals that have common properties and belong to the same group. OWL classes are the most basic concept that are the roots of various taxonomic trees. Every individual in the OWL document is a member of the *owl:Thing* class. Thus, each created class is implicitly a subclass of *owl:Thing*. Domain specific root classes are defined by simply declaring a named class. OWL also defines the empty class, *owl:Nothing*.

Figure 2.3 shows two declarations of root classes inside an OWL ontology. OWL classes are defined inside an element *< owl : Class >*. The declarations shown above describes only the unique ID of the classes, without going deeper. A class can be defined as the union, intersection and complement of other classes by using the constructs *owl:unionOf*, *owl:intersectionOf* and *owl:complementOf* respectively, or as an enumeration of its members by using the construct *owl:oneOf*.

Moreover, the most specific component of the classes is *rdfs:subClassOf*. It connects a more

---

```
1 <owl:Class rdf:about="RedWine">
2   <rdfs:subClassOf rdf:resource="#Wine"/>
3   ...
4 </owl:Class>
```

---

Figure 2.4: OWL subclass definition

particular class with a more general one. The *rdfs:subClassOf* relation is derivative, if X is a subclass of the class Y, then every instance of X is also an instance of Y. The *rdfs:subClassOf* relation is also transitive, so that if X is a subclass of class Y and Y a subclass of class Z then X is a subclass of Z. Moreover, OWL class has some more descriptions to extend the definition of a resource. For example, see the declaration of *rdf:about* in Figure 2.4.

Figure 2.4 shows how the class *RedWine* is derived from the general class *Wine*. The construct *rdf:about* is used because the class *RedWine* is already declared and at this moment we want to extend this class by relating it to a general class, through the subclass mechanism, in order to inherit the properties and the characteristics of *Wine*. Furthermore, two OWL classes may be regarded as equivalent or disjoint by using the mapping constructs *owl:equivalentClass* and *owl:disjointWith*, respectively.

**OWL individuals** are the instances of classes, see example in Figure 2.5. Instances are declared by using the *rdf:type* construct or the name of the class as the name of the element in which the individual is defined. The individuals may have the properties and have to satisfy all the constraints that are predefined for the corresponding OWL class.

**OWL properties** provide general facts about the classes and specific facts about the class individuals. There are two categories of properties: object properties and data type properties.

---

```

1      <RedWine rdf:ID="Syrah">
2
3      -OR-
4      <owl:Thing rdf:ID="Syrah"/>
5      <owl:Thing rdf:about="#Syrah">
6          <rdf:type rdf:resource="RedWine"/>
7      </owl:Thing>

```

---

Figure 2.5: OWL class individual

---

```

1      <owl:Class rdf:ID="VintageYear">
2      <owl:DatatypeProperty rdf:ID="yearValue">
3          <rdfs:domain rdf:resource="#VintageYear"/>
4          <rdfs:range rdf:resource="xsd:positiveInteger"/>
5      </owl:DatatypeProperty>

```

---

Figure 2.6: OWL Datatype property definition

- **Object properties** are relations between the instances of two classes. An object property is described using the *owl:objectProperty* construct, which connects individuals of the domain class with individuals of the range class.
- **Data type properties** are relations between class instances and RDF literals or XML Schema data types. A data type property is defined by using the *owl:DatatypeProperty* construct, which relates individuals of the domain class to values of the range data type. The example of data type property is illustrated in Figure 2.6.

Figure 2.6 describes the definition of a data type property which relates the *vintage* years of a *wine* production to positive integers. An instance of the *VintageYear* class is shown in Figure 2.7.



---

```

1      <VintageYear rdf:ID="Year1998">
2          <yearValue rdf:datatype="&xsd:positiveInteger"
3              >1998</yearValue>
4      </VintageYear>

```

---

Figure 2.7: OWL Class instance with datatype property

#### 2.1.4 Term definitions

Since our thesis usually use the term *structure* and *semantics*, in this section we restate their definitions again. According to the business dictionary [30], *structure* is the construction of identifiable elements in which each element is functionally connected to others, and the interrelationships between elements are fixed or changing occasionally or slowly. Based on this definition, we can infer that XML element's structure is the relation of that element to its *ancestor*, *sibling*, and *descendant* elements. Therefore, the structure similarity of XML element is the combination of the similarity scores of those relation elements.

According to the Kamil [99], *semantics* is the scientific study of the meaning of words. This meaning is analyzed in terms of their semantic features which are the way that a word is used in a document. From this definition, we figure out that semantic similarity between XML elements is the combination of the *meaning* similarity of element name and the similarities of their other characteristic, such as *data type*, *cardinality constraint*.

## 2.2 Related Work

As mentioned in the previous chapter, our goal is to enhance the data sharing between XML applications by integration of XML Schemas (XSDs) and transformation of XML schema (XSD/DTD) into OWL ontology based on the similarity measures. To perform these two tasks, it is required to measure the similarity of elements in XML schemas. The main difference of similarity measure in two methods is: The first method is based on the similarity measure of elements in two different documents, whereas the second method relies on the similarity measure of elements within a single document. Therefore, in this section, we introduce two subsections: XML integration with element similarity between different documents and XML transformation with element similarity within a document.

### 2.2.1 Similarity between documents and XML integration

Much work has addressed the similarity between XML documents. Similarity can be computed at different layers of abstraction: at the instance layer (i.e., similarity between instance documents), at the schema type layer (i.e., similarity between data types, also referred to as schema, models, or structures, depending on the application domain), or between the two layers: instance and schema. XML similarity can be categorized as either of three approaches: (1) structural similarity or (2) semantic (content) similarity or (3) Hybrid approach: semantic (content) and structural similarity.

#### 2.2.1.1 Structural similarity

Structural similarity focuses mainly on the relationship similarity of elements between schema graphs. David Buttler [14] summarized three approaches to structural similarity: (1) tag similarity, (2) tree edit distance (TED), and (3) Fourier transform similarity.

##### **Tag similarity**

This is the most simplest way to measure the structural similarity between XML documents. It

measures how close element names from the two XML documents are. Documents which use resemblance element names are likely to have similar schema. This measure evaluates the number of intersected elements from the compared documents and it is divided by the union of elements between two documents. However, this approach is not suitable for several reasons. One critical problem is that some XML documents deriving from the same schema may have only a limited number of element names, whereas some XML documents may contain a large number of a particular element name. In addition, tag similarity completely ignores the similarity of the relationships between elements, thus yielding low similarity quality.

### **Tree edit distance (TED)**

According to Bille [9], *tree edit distance* between two labeled trees,  $T_1$  and  $T_2$ , is the optimal sequence edit operations that turn  $T_1$  into  $T_2$ . The edit operations include of insertion, deletion, and substitution. Previously, those edit operations are only applied on single nodes. One of the typical approach is Chawathe's method [17]. They performs the insertion and deletion operations at the leaf-node level and process the substitution of node labels anywhere in the tree but, without considering the move operation. The overall complexity of Chawathe's algorithm is expressed as  $O(N^2)$  where  $N$  is the maximum number of nodes of the compared trees. This complexity is quite expensive then leads to the longer run time. Therefore, Chawathe's approach is not practical for measuring the similarity of large XML data.

On the other hand, one of the typical approach, which uses the complex edit operations is proposed by Shasha et al. [103]. They introduce a TED metric that permits the addition and deletion of single leaf node anywhere in the tree, not just at the leaf level. However, the entire subtrees cannot be inserted or deleted in one step. The complexity of this approach is expressed as  $O(|T_1||T_2|depth(T_1)depth(T_2))$ . Here,  $|T_1|$  and  $|T_2|$  represent the number of nodes in label trees  $T_1$  and  $T_2$ , respectively.

Nierman and Jagadish [69] focuss on the structural similarity of the subtrees. Their edit operations are similar to Chawathes, but they add two more new operations: insert tree and delete tree. To determine subtree similarities, they introduce containment in the relationship between trees or subtrees. A labeled tree  $T_1$  is said to be contained in a labeled tree  $T_2$  if all nodes of  $T_1$  occur in  $T_2$  with the same parent/child edge relationship and node order. The overall complexity of this algorithm is expressed as  $O(N^2)$ . This approach proved more accurate in detecting XML structural similarities than those of either Chawathe or Shasha.

Also based on Chawathe's method, Dalamagas et al. [23] introduce a framework for clustering XML documents on the basis of the structure similarities. They present the XML documents as rooted ordered labeled trees, then study the usage of structural distance metrics in hierarchical clustering algorithms to detect groups of structurally similar XML documents. Wei Li et al. [51] extend Dalamagas' method to cluster dynamic XML documents based on the frequently changing in their structures.

There are other three approaches which are based on structural similarity but result in higher accuracy than TED method. First, Lian et al. [53] represent XML document structures as directed graphs called s-graphs, and define a distance metric that captures the number of edges common to the graph representations of two XML documents:

$$Dist(G_1, G_2) = \frac{1 - |Edges(G_1) \cap Edges(G_2)|}{MaxEdges(G_1), Edges(G_2)} \quad (2.1)$$

This equation 2.1 is more effective than others based on TED, in separating documents that are structurally different. It can be applied not only to tree-structured documents but also to document collections of arbitrary (graph) structure.

Second, Bertino et al. [8] proposed a matching algorithm for measuring the structural sim-

ilarity between an XML document and a DTD. By comparing the document structure with that required by the DTD, the matching algorithm is able to identify commonalities and differences. Differences can be the occurrence of extra elements beyond those required by the DTD, or the absence of required elements. The degree of similarity can be evaluated based on the elements properties, such as level or weight. Elements at higher levels are considered more relevant than those at lower levels. The authors state that their approach is of exponential complexity.

Third, Rafiei et al. [78] use the path expressions of XML documents to find the similarity among these documents by measuring the similarity of paths between them. A path is defined as a list of connected nodes starting at the root and terminating at a leaf node. Rafiei et al. [78] conclude that two XML documents are similar if they share a large fraction of the paths in their path sets. The path set includes all root paths (from the root to leaf nodes) and all possible subpaths. The time complexity in terms of the number of string comparisons is expressed as  $O(nl^2)$ , where  $n$  is the number of root paths and  $l$  is the length of each path. The survey of Buttler [14] shows that the path similarity method provides fairly accurate results compared to TED.

### **Fourier transform similarity**

Essentially, Fourier transform similarity [78] removes all the information from a document except for its start and end tags, leaving only its skeleton, which represents its structure. The structure is then converted into a sequence of numbers, which is viewed as a time series, and a Fourier transform is applied to transform the data into a set of frequencies. Finally, the distance between two documents is computed by calculating the difference in the magnitudes of the two signals. Buttler [14] proved that this algorithm is the least accurate of all approximation algorithms, and performs poorly because Fourier transform does not differentiate sufficiently between very similar documents.

In general, the structural similarity approaches mainly concentrate on the distance similarity of elements in the tree model whereas they do not consider the meaning (or the semantics) of element names. Moreover, according to the definition of *structure* in the Section 2.1.4, the structure similarity is the combination of the similarity of ascendant, sibling, and descendant elements. However, there is no approach presented in this structure similarity section obtains all those features. Our purpose is to include all the relationships of the XML elements in computing their similarity.

### 2.2.1.2 Semantic similarity

Semantic similarity plays an important role in information systems as it supports the identification of objects that are conceptually close but not identical. Semantic assessment is particularly significant in different areas of knowledge management (such as data retrieval, information integration, and data mining) because it facilitates the comparison of the information resources in different types of domain knowledge. In this section, we analyze different approaches in discovering and ranking semantic similarity between concepts in different documents.

The simplest approach to semantic similarity is a linguistic-based metric that computes similarity between names or descriptions of two elements by using string matching [24]. There are a variety of string matching algorithms, such as the widely used Jaccard [83] and Cosine similarity [29] measures. Others have proposed methods based on a linguistic taxonomy [102] such as WordNet [94], from which user can obtain more accurate and less ambiguous semantics for words in the element names.

The string-based similarity metrics can be enhanced by using natural language preprocessing

techniques, such as tokenization, lemmatization, and elimination [39]. To further enhance the string-based metrics, common knowledge repository, such as WordNet [94] and domain-specific corpora, can be used to determine the meaning of the words. Based on the knowledge repository, several methods have been proposed [76], [102].

In another approach, Jeong et al. [40] propose a semantic similarity measuring method which considers data instances as well as schema-based information. The instance-level data give important insight into the contents and meaning of schema elements. The main advantage of this approach is that it can obtain the similarities among data, relying on their instances. To perform this method, Jeong et al. use neural network-based partial least squares to measure the semantic similarity of XML elements based on their instances. However, the main disadvantage of using instance-based approach is that instance data is generally available in very vast quantity. Hence, the computational cost is very expensive, which affects the similarity scores.

Moreover, according to the definition of semantics in Section 2.1.4, the semantic similarity of XML elements is the combination of meaning similarity of element names and other element features, such as data type and cardinality constraint. However, none of above presented semantic similarity measuring methods compute other features of XML elements.

In general, presented methods above only focus on measuring one similarity factor, either structural or semantic similarity, therefore the combination of both method, called **Hybrid Similarity**, is necessary. Semantic similarity plays an important role in finding the similarity in meaning or semantic content. Structural similarity measures have performed strongly with resources containing large amounts of text, but they cannot appropriately cope with structural and semantic heterogeneity and ambiguity if the semantics of the terms are not explicitly available. The hybrid similarity measure combines both semantic and structural similarity measures to detect the simi-

ilarity among documents. This approach can be incorporated using average, maximum, additive, or weighted sum functions. The average, maximum and additive functions are simple. The weighted sum seems to work best, but it requires that the domain experts obtain the weight values.

### 2.2.1.3 Hybrid: Similarity of XML structure and semantics

In the context of XML classification and clustering, structural similarity seems to be sufficient to distinguish or classify XML documents. However, in the context of *XML data integration*, not only the structure of XML documents must be considered, but also their content in order to determine whether XML documents have similar content to integrate.

A typical hybrid approach is from Kade and Heuser [43]. They propose an approach called XSIM that uses information from both structure and content of XML documents. Three pieces of information permit calculation of the similarity between two nodes of XML trees: the content of the element and the names and path of the nodes. The comparison has two main steps: (1) node matching and (2) document matching. First, the document tree is traversed to produce a set of tuples containing the path and content. This step results a document list for subtrees. Second, the tuples of the document lists in the first step are compared and based on the similarities of the textual content, the node label and the node path. Finally, the similarity between two elements of two XML documents is computed as the average of textual content, element name, and path similarity values.

Another approach that follows a probabilistic model uses a Bayesian network [28]. This approach computes the similar probabilities of ascendants and descendants in a given pair of XML subtrees. To compare two XML elements, a maximum overlay between the two trees is computed. Two nonleaf nodes can be matched if they are ancestors of two matched leaves. Once a maximum



overlay has been determined, its cost is computed by a string distance function.

Leito, Calado, and Weis [50] also use Bayesian network to compute the similarity of XML elements. They prove that the model provides great flexibility in its configuration, allowing the use of various similarity measures for the field value and various conditional probabilities to combine the similarity probabilities of the XML elements. However, the primary disadvantage of Bayesian techniques is their computational complexity.

Other effective approaches are proposed by Nayak et al. [66], [67], and Algergawy et al. [2]. We name those approaches are *weight-based methods*. They employ supplemental functions to calculate the similarity of a particular feature of the given schema, such as the similarity of leaf nodes or root nodes, and similarity of data type and constraints. All the partial results are then combined into the final similarity value using a weighted sum function. Brauner et al. [13] describe a matching algorithm based on measuring the similarity between the property domains of distinct Web databases. Madhavan et al. [55] propose the use of a set of schemas and mappings to help the schema matching algorithms. The authors use predictor algorithms that measure the similarity between schema elements, adopted in the PayGo [38] architecture [56].

Our similarity measuring method is more related to weight-based methods. Nevertheless, most of them focused only on measuring the name similarity [101], [24], [102], or on the structural similarity of the elements between two documents [89]. Some studies concentrated on both semantic and structure, but some factors in their metrics should be assigned manually using human's judgement [47], [2]. Moreover, the XML data are very large and contain varied definitions for data types and cardinality constraints, they need formal metrics to obtain the accurate similarity values.

### 2.2.2 Duplicate similarity and XML schema transformation

In this section, we first describe the related work to XML transformation together with the duplicate problem during the transformation process. To solve the duplicate problem we should measure the similarity of duplicate in a schema. Therefore, we describe the related work to measure the similarity of concepts within a document in the next section.

#### XML schema transformation

Since great amount of data is stored in XML and one idea of the Semantic Web is to enable transparent interoperability between these data sources, there are several studies of XML data to OWL migration. These approaches differ in purpose, architecture, usability, algorithm efficiency, and so on. In this section, we examine major contributions to this field and their main attributes.

There are several approaches which focus on the schema mapping and transformation have been proposed. One of the typical approaches is from Ferdinand et al. [26]. They describe the transformation of XML schema into RDF and mapping of XML schema to OWL ontology. However, the mapping rules are defined by users so that OWL results could not be used directly by the Semantic Web applications since the confliction of OWL syntaxes. On the other hand, some approaches employ XSLT (eXtensible Stylesheet Language Transformations) to transform XML into OWL, such as Hannes et al. [10] and Tsinaraki et al. [92]. Although those approaches have the same purpose in XML data transformation, their solution are different. Particularly, to solve the duplicate problem of XML data, Hannes et al. add prefixes such as *has* and *dtp* before class and property elements, respectively, whereas Chrisa et al. concatenate the ancestor names with the current element name. Similarly, Bernd et al. [3] attached the key paths to each nested element, whereas Toni et al. [82] and Cruz et al. [22] used XPath [97], [57], [7] expressions to express the XSD elements. These techniques prevented XML duplicates from occurring in the OWL ontology, but they altered most of the XML element names. Our previous approach [91] only changed the

name of duplicate elements by adding their corresponding ancestor name, but we did not consider the duplicate similarity. In another approach, Amann et al. [3] attach the *key paths* to each nested element. This method gives more semantics for the source document, but it requires the user's intervention. Similarly, two approaches [82], [22] use XPath expressions to translate the XML Schema elements.

In general, approaches related to transforming XML into OWL ontology assume that all XML elements are different from each others and the researchers can assign each element a differentiating ID, whereas approaches related to measuring the similarity of concepts within a single document assume that if two elements have the same name, they are similar to each other. However, in real-world schemas, some duplicate elements are similar but others are non-similar. Therefore, to ensure the correct semantics of ontology, we propose method to measure the similarity of duplicates in XML schemas before doing the transformation them into OWL ontologies. Before describing our duplicate similarity method, we introduce some related approaches which measure the similarity of element within a document.

### Similarity within a document

Discovering similarity between nodes in a taxonomy has a long story in literature. There are several researchers using direct relations between nodes, based on either structural similarity or semantic similarity of node names. Their approaches can be classified into three main groups:

1. **Edge-based** that considers the number of edges between nodes but does not rate different characteristics for each edges
2. **Node-based** that is more accurate but needs more computation for the content of each concept
3. **Hybrid**: the combination of both edge-based and node-based methods.

There are several approaches which are related to the element similarity measurement in a single document. One of the most famous method is called *Edge-Counting metrics* [80], [87]. This edge-based approach is a simple and intuitive way of evaluating the similarity of concepts in a taxonomy. This approach estimates the distance between nodes corresponding to the concepts being compared. On the other hand, Rada et al. [77] and Leacock et al. [46] show that the simplest means of determining the distance between two concept nodes, A and B, is identifying the shortest path that links A and B, or the minimum number of edges that separate A and B.

In another approach, Jiang and Conrath [42] state that the distance between any two adjacent nodes is not necessarily equal; therefore, this approach is not sensitive to the problem of varying link distances. Edge weight can be considered in order to solve this problem. It is related to the number of children, the depth of a node in the hierarchy, the type of link (such as the is-a, part-of, or substance-of links), the network density, and the strength of an edge link.

Moreover, there is another method which measures the depth of two concepts in a taxonomy and the depth of the least common subsume (LCS) [19], [5]. It then combines these properties into a similarity score:

$$sim_{Wu}(c_1, c_2) = \frac{2 \times depth(LCS(c_1, c_2))}{depth(c_1) + depth(c_2)} \quad (2.2)$$

where LCS is the lowest common subsume and is the depth of node c in the hierarchy.

Some other research groups [54], [81] are based on the information-content (IC) method [45] to measure the semantic similarity between terms within a taxonomy. These measures were designed mainly for WordNet [94]. For instance, Resniks measure [81] calculates the semantic similarity between two terms  $[t_1, t_2]$  in a given ontology (e.g., WordNet) as the information content (IC) of the least common ancestor (LCA) of  $t_1$  and  $t_2$ . The IC of a term  $t$  can be quantified in terms of the probability ( $P(t)$ ) of its occurrence. The probability assigned to a term is defined as

its relative frequency of occurrence:

$$sim_{Resnik}(c) = IC(c) = -\log P(c) = -\log P(LCS(c_1, c_2)) \quad (2.3)$$

where  $P(c)$  is the probability that a randomly selected word in a corpus is an instance of concept  $c$ . This can also be written as:

$$P(c) = \frac{\sum_{w \in words(c)} count(W)}{N} \quad (2.4)$$

where  $words(c)$  is the set of words subsumed by concept  $c$ , and  $N$  is the total number of words in the corpus.

Another widely known method to assess the similarity between two concepts is called **String matching** [16]. It is important in the domain of text processing. It involves identifying a place where one or several strings are found within a text. String matching algorithms generally scan the text with the help of a window. They first align the left ends of the window and the text, then compare the characters of the window (called patterns). After a whole match of the pattern, or after a mismatch, they shift the window to the right. The same procedure is repeated until the right end of the window goes beyond the right end of the text. This mechanism is called the *sliding window mechanism* [35].

In general, approaches which measured the similarity of concepts within a document almost focus on the string name of elements and the distance comparison of those elements with their ascendant and descendant elements. Moreover, most approaches assume that duplicate concepts have highest similarity score (1) such as Leacock and Chodorow [46], Li et al. [52], and Resnik [81]. In contrast, in our approach, duplicate concepts may have different similarity scores depending on their inside declarations (name, data type, and cardinality constraint) and outside relationships (ancestor, sibling, and children).

## Chapter 3

# ESim: Element Similarity measure for XML integration

In this chapter, we introduce the general framework of the similarity measure method, and then present the motivation example and the details of ESim method.

### 3.1 Similarity Measure Framework

In this section, we describe a general framework for measuring the element similarity. Note that this framework is used for both measuring scenarios: element similarity among different documents and element similarity (specifically, duplicate similarity) within a document. This framework is built based on the semantics and structure definitions in Section 2.1.4. Depending on each scenario, the measuring metric of each factor is different. The general similarity measuring framework is present in Figure 3.1.

The general framework of similarity measuring method includes the the input, the similarity computation, and the output. The input is a schema file (XML Schema Definition (XSD) or Doc-

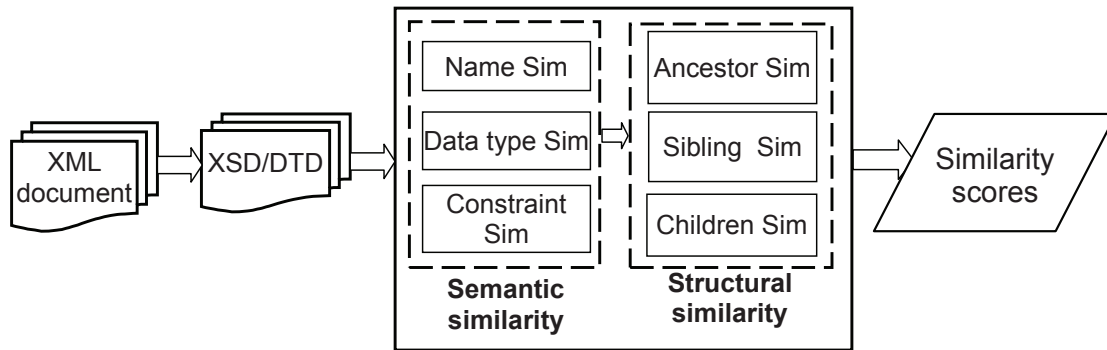


Figure 3.1: General framework of similarity measure method

ument Type Definition (DTD)), which usually comes with its XML document. Sometimes, an XML document appears alone without any schema. In this case, its schema can be extracted using HIT software [6]. Since XSD is the extendable format of DTD, in this thesis our descriptions concentrate on XSD, and we introduce how to apply our approach to a DTD document. The main component of this framework is the similarity computation, which is composed of the semantic and structural similarity measures. The output are the similarity scores of element pairs among XSD documents or the duplicates within a XSD document.

The semantic similarity in Figure 3.1 comprises the similarity of the element name (Name Sim), the data type compatibility (Data type Sim), and the cardinality constraint similarity (Constraint Sim). Similarly, the structural similarity encompasses three individual element measures: the ancestor similarity (Ancestor Sim), the sibling similarity (Sibling Sim), and the children similarity (Children Sim). The final similarity measure is the combination of all the partial results using a weighted sum function. Details of each measuring metric are presented in next section, and next chapter.

To illustrate for our method, we first restrict ourselves to hierarchical schemas. Thus, we model

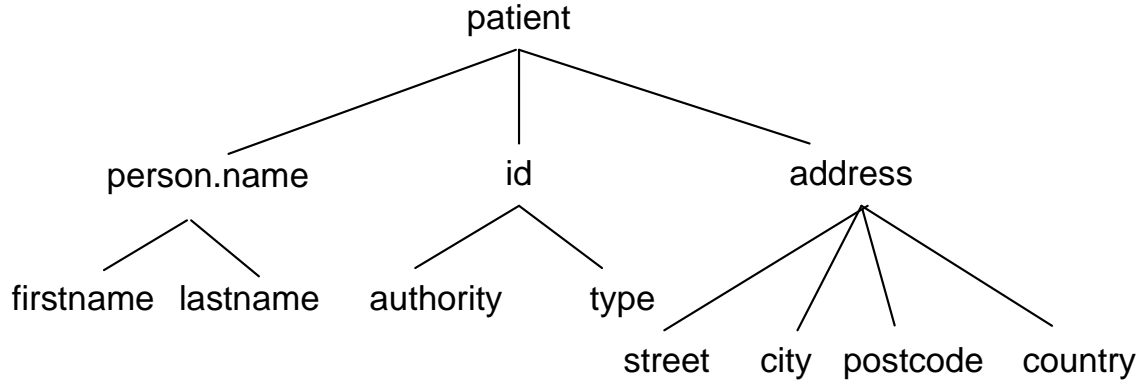


Figure 3.2: Tree representation for Schema Patient\_A

the interconnected elements of an XML Schema as a schema tree. We use two XML Schema trees illustrated in Figures 3.2 and 3.3 below for explaining our algorithm.

Figure 3.2 and Figure 3.3 show two XML Schema trees for different XML Schemas Patient\_A and Patient\_B for representing patients, respectively. We would like to match these two XML Schemas. Both XML Schemas have the same name, *patient*. The XML Schemas are encoded as graphs, where the nodes represent schema elements. Although even a casual detector can see that both schemas are quite similar, there is still much variation in naming and structuring that make the matching algorithm being challenged.

In this chapter, we compute the similarity coefficients between elements of the two schemas and then deducting an integration from those coefficients. The coefficients are calculated in two stages. The first stage, semantic similarity measure, compares individual elements based their names (linguistics), data type, and cardinality constraint similarities. The second stage computes the structural similarity. To handle the abbreviation of names (linguistic similarity), we use Word-Net [94] to determine whether these names are synonym or not. The similarity between elements is declared as following Definition 3.1:



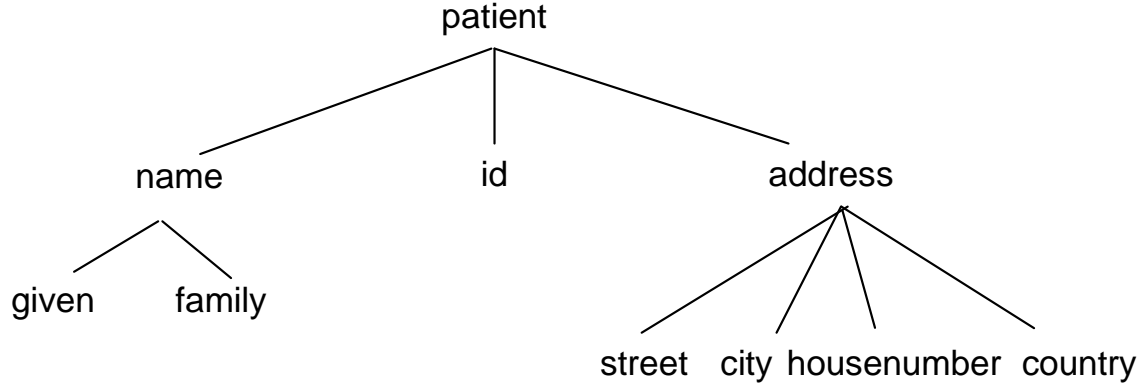


Figure 3.3: Tree representation for Schema Patient\_B

**Definition 3.1:** The element similarity (*ESim*) between  $e_1$  and  $e_2$  is defined as the weighted sum of the semantic similarity (*SeSim*) and structural similarity (*StSim*):

$$ESim(e_1, e_2) = \alpha * SeSim(e_1, e_2) + (1 - \alpha) * StSim(e_1, e_2) \quad (3.1)$$

where the coefficient  $\alpha$  is a weight parameter, a constant between 0 and 1. The weight factor  $\alpha$  is used to scale the ESim results to range between 0 and 1. Higher ESim values represent greater similarity between elements. According to experiments, ESim has close value to users perspective at  $\alpha = 0.55$ . Details of each similarity measurement are presented in the next sections.

## 3.2 Semantic Similarity Measurement (SeSim)

As introduced in the previous section, our semantic similarity composes of similarities of element name, data type, and cardinality constraint. The reason is that according to the definition of *semantics* in Section 2.1.4, semantic similarity is the similarity of element meaning and other characteristics of the elements. Therefore, the semantic similarity of elements in one type of doc-

ument (such as XML Schema) is different from another (such as relational database, web page, etc.). Since our thesis focuses on similarity of elements in XML Schema, we explore the characteristics of XML Schema.

The main characteristics of XML Schema include the vocabularies, the content model, the data type, and the cardinality constraint. The vocabulary, which contains element and attribute names, allows us to determine the name similarity between elements in the XML Schema. The content model, which presents the relationship and structure of the elements, is used to compute the context similarity and cardinality constraint similarity of the duplicate elements. The cardinality constraint defines the number of possible occurrences for an element by the *maxOccurs* and *minOccurs* attributes. The data type, which defines the data types for elements and attributes, helps us to measure the similarity between their data types. Usually, XML Schema uses the standard namespace, (*xs* or *xsd*), and the *URI* associated with this *namespace* to begin the document. Moreover, *simpleType* or *complexType* element helps us to differentiate the data type similarity between two attribute types of the elements. For instance, the data type similarity between simple and complex elements is zero. Figure 3.4 presents XML Schema expressions for Patient\_A in Figure 3.2.

Based on the above analysis of XML Schema characteristics, we propose the metric to measure the semantic similarity of XML element by the following Definition 3.2.

**Definition 3.2:** The semantic similarity (*SeSim*) between two elements  $e_1$  and  $e_2$  is defined as the weighted sum of the name similarity (*NSim*), data type similarity (*DSim*), and the constraint similarity (*CSim*) as follows:

$$SeSim(e_1, e_2) = \beta * NSim(e_1, e_2) + \chi * DSim(e_1, e_2) + (1 - \beta - \chi) * CSim(e_1, e_2) \quad (3.2)$$

where  $\beta$  and  $\chi$  are the weight factors,  $0 \leq \beta, \chi \leq 1$ . According to our experiment, at the values of

---

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3  <xs:element name="patient">
4      <xs:complexType> <xs:sequence>
5          <xs:element ref="person.name"/>
6          <xs:element ref="id" maxOccurs="unbounded"/>
7          <xs:element ref="address"/>
8      </xs:sequence> </xs:complexType>
9  </xs:element>
10 <xs:element name="person.name">
11     <xs:complexType> <xs:choice>
12         <xs:element ref="firstname"/>
13         <xs:element ref="lastname"/>
14     </xs:choice> </xs:complexType>
15 </xs:element>
16 <xs:element name="id">
17     <xs:complexType mix="true">
18         <xs:attribute name="type" type="xs:NMTOKEN" use="optional"/>
19         <xs:attribute name="authority" type="xs:NMTOKEN" use="optional"/>
20     </xs:complexType> </xs:element>
21     ...
22 </xs:schema>
```

---

Figure 3.4: Expressions for Schema Patient\_A

$\beta = 0.4$ ,  $\chi = 0.3$ , similarity score proposed by our method returns close results to users perspective. Each similarity measure in Equation 3.2 is presented in the following subsections.

### 3.2.1 Name similarity (NSim)

The most important factor of the element measurement is the name similarity. The name similarity computes the linguistic and semantic similarity between two elements within a schema. Element names in the XML Schema are often declared as a word or a set of words. Moreover, since XML tags are created freely, similar semantic notions can be represented by different words (e.g., medicine and drug), or different elements can have linguistic similarity (e.g., cure and cured). The name similarity between elements is computed by three main steps. The first step normalizes each element name to remove genitives, punctuation, capitalization, stop words (such as, of, and, with, for, to, in, by, on, and the), and inflection (plurals and verb conjugations). After normalizing the element name, the first step separates the composed element into single words. For example, *HeartOfPatient* becomes *heart* and *patient*.

The second step finds the synonyms for each compared element name by looking them up in the WordNet thesaurus [94] and then computes the name similarity between elements. We reuse the distance based method [101] to compute the semantic similarity of elements  $e_1$  and  $e_2$  by referring them in WordNet. The name similarity is determined by following Equation 3.3:

$$NSim(e_1, e_2) = \frac{2 * depth(LCS)}{depth(e_1) + depth(e_2)} \quad (3.3)$$

where  $depth(LCS)$  is the number of nodes from the common super-concept of  $e_1$  and  $e_2$  to the root node;  $depth(e_1)$  and  $depth(e_2)$  are the numbers of nodes from  $e_1$  and  $e_2$  to the root node.

For instance, given a fragment of WordNet in Figure 3.5, the name similarity between elements *lecturer* and *professor* is:  $NSim(lecturer, professor) = \frac{2*4}{5+6} = 0.73$

In the case that one of the two elements is processed by the tokenization step, then the name

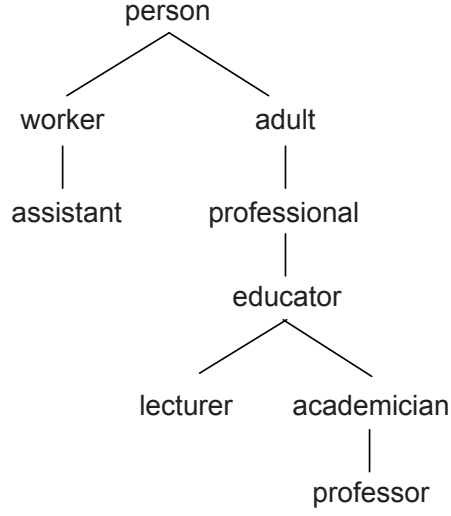


Figure 3.5: A fragment of WordNet

similarity of these two elements is presented as in matrices 3.4 and 3.5:

$$NSim(E_1, E_2) = \begin{bmatrix} NSim(e_{1_1}, e_{2_1}) & \cdots & NSim(e_{1_1}, e_{2_n}) \\ \vdots & \ddots & \vdots \\ NSim(e_{1_m}, e_{2_1}) & \cdots & NSim(e_{1_m}, e_{2_n}) \end{bmatrix}, m \geq n \quad (3.4)$$

$$NSim(E_2, E_1) = \begin{bmatrix} NSim(e_{2_1}, e_{1_1}) & \cdots & NSim(e_{2_1}, e_{1_m}) \\ \vdots & \ddots & \vdots \\ NSim(e_{2_n}, e_{1_1}) & \cdots & NSim(e_{2_n}, e_{1_m}) \end{bmatrix}, m < n \quad (3.5)$$

where  $m$  and  $n$  are the number of words in the token sets of the elements  $E_1$  and  $E_2$ , respectively;  $NSim(e_1, e_2)$  is the name similarity between elements  $e_1$  and  $e_2$ , determined by equation 3.3. The name similarities of two elements  $E_1$  and  $E_2$  in the matrices 3.4 and 3.5 are determined by either following equation 3.6 or 3.7, respectively:

$$NSim(E_1, E_2) = \frac{\sum_{i=1}^m \max_{j=1}^n (NSim(e_{1_i}, e_{2_j}))}{m}, m \geq n \quad (3.6)$$

$$NSim(E_2, E_1) = \frac{\sum_{i=1}^n \max_{j=1}^m (NSim(e_{2_i}, e_{1_j}))}{n} \quad m < n \quad (3.7)$$

where  $\max$  is the maximum similarity value of each row in the matrices.

In some cases, the compared element names are not in WordNet, we define metric (Equation 3.8) to measure the name similarity in that case:

$$NSim(e_1, e_2) = LingSim(e_1, e_2) = \frac{n_{e_1 \cap e_2}}{\max(n_{e_1}, n_{e_2})} \quad (3.8)$$

where  $n_{e_1 \cap e_2}$  is the number of matching characters between elements  $e_1$  and  $e_2$ ;  $\max$  is the maximum value;  $n_{e_1}$  and  $n_{e_2}$  are the lengths of the elements  $e_1$  and  $e_2$ , respectively.

### 3.2.2 Data type similarity

Although the main factor of semantic similarity calculation is the element name, the consideration for other components also plays a very important role. For instance, the name similarity value between two elements *id* in Figure 3.2 and Figure 3.3 is 1. However, this is a false matched value, since the first *id* element is a complex element whereas the second *id* element is a simple element. This means they are different in other attributes. Therefore, it is necessary to use other factors to calculate their semantic relatedness to eliminate some false matches.

In the XML Schema document, every element is always either simple or complex type. If two elements have the same name, and their data type properties are identical (both are a complex type or simple type), their semantic similarity may be higher than other cases, such as simple and complex. Since the complex element contains children, in order to compute the similarity between two complex elements, we have to compare the similarity of their children. This problem is mentioned in the structure similarity measurement section. For two leaf elements, we concern their data type declaration. Since the data type often comes with an attribute element, the data

type measurement is only applied for the case that both elements are attributes. In the case that two elements are complex types or the first element is a complex type, the second is the simple type, the data type similarity is 0. For example, the element *id* in Figure 3.2 has a complex type, and element *id* in Figure 3.3 has a string data type. Therefore, the data type similarity between two elements (complex, string) is 0.

Based on the summary about XML Schema data types [75], [41], we divide them into 12 data types which are presented in Table 3.1. Some of them represent a set of communal data types. The details of the 12 data types are as follows:

1. **URI** includes *anyURI* and *base64Binary*.
2. **Lang** consists of *language* and *list*.
3. **Text** includes text characters, such as *normalizedString*, *string*, and *token*.
4. **Ubyte** is all positive whole numbers, such as *unsignedByte*, *nonNegativeInteger*, *positiveInteger*, *unsignedInt*, *unsignedLong*, and *unsignedShort*.
5. **Decimal** contains all decimal number data types, such as *decimal*, *double*, and *float*.
6. **Int** contains both positive and negative whole number: *byte*, *int*, *integer*, *long*, *nonNegativeInteger*, *nonPositiveInteger*, *positiveInteger*, *short*, *unsignedByte*, *unsignedInt*, *unsignedLong*, and *unsignedShort*.
7. **DTime** is composed of a set of data types, such as *date*, *dateTime*, *duration*, *gDay*, *gMonth*, *gMonthDay*, *gYear*, *gYearMonth*, and *time*.
8. **Name** comprises *Name*, *NCName*, *NOTATION*, and *QName*.
9. **Entity** consists of *ENTITY* and *ENTITIES*.
10. **ID** comprises *ID*, *IDREF*, and *IDREFS*.

11. **Token** includes *NMTOKEN* and *NMTOKENS*.

12. **Type** contains *anyType* and *anySimpleType*.

We find that other approaches related to measuring the similarity of data type, such as [2], [66], often assign the similarity value for each data type pair, and they only mention some common data types of the XSD. In this work, we introduce a novel technique to calculate these values. Based on the constraining facets of XML Schema [75], we define the metric for measuring the similarity among the data types. The Data type similarity of two element  $e_1$  and  $e_2$  is the fraction of common number constraining facets per the maximum (*max*) number of constraining facets of each element :

$$DSim_1(e_1, e_2) = \frac{\sum_i |\{cf_i | e_1[cf_i] = e_2[cf_i], 1 \leq i \leq n_{cf}\}|}{max(\#e_1.cf, \#e_2.cf)} \quad (3.9)$$

where  $DSim_1$  is the data type similarity based on the resemblance of constraining facets;  $cf$  is one of the constraining facets described in [75],  $max(\#e_1.cf, \#e_2.cf)$  is the maximum number of constraining facets of the data type of the element  $e_1$  and  $e_2$ .

The results of equation 3.9 are quite acceptable except for some illogical values. For instance, the resemblance of *date* and *float* is 1.0, and the similarity between *decimal* and *integer* is also 1.0, although the number of constraining facets between *date* and *decimal* is different. Instead, we expect that those similarities values are less than 1.0, and the similarity between *decimal* and *integer* is higher than that of *date* and *float*.

Thus, we insert another metric to measure the data type similarity based on the number of constraining facets of each data type over the total number of constraining facets. This technique is names  $DSim_2$ , and it is determined by the following equation:

$$DSim_2(e_1, e_2) = \frac{max(n_{e1.cf}, n_{e2.cf})}{n_{cf}} \quad (3.10)$$



where  $\max(n_{e1.cf}, n_{e2.cf})$  is the maximum number of constraining facets of the data type of the element  $e_1$  and  $e_2$ ;  $n_{cf}$  is the number of constraining facets, in this case  $n_{cf} = 12$ . The constraining facets include *length*, *minLength*, *maxLength*, *pattern*, *enumeration*, *whiteSpace*, *maxInclusive*, *minInclusive*, *maxExclusive*, *minExclusive*, *totalDigits*, and *fractionDigits*.

The combination of *DSim1* and *DSim2* produces the data type similarity, *DtSim*, of two elements  $e_1$  and  $e_2$ . *DtSim* is measured by the following definition:

$$DSim(e_1, e_2) = \frac{\delta_1 * DSIM_1(e_1, e_2) + \delta_2 * DSIM_2(e_1, e_2)}{\delta_1 + \delta_2} \quad (3.11)$$

where  $\delta_1$  and  $\delta_2$  are weight parameters between 0 and 1. In this paper, we assign 0.5 to  $\delta_1$  and  $\delta_2$ , because we assume that *DSim1* and *DSim2* have similar roles. With equation 3.11, we can moderate the results of data type similarity. The final data type similarity, *DSim*, among some common XSD data types are presented in Table 3.1.

Table 3.1 presents the data type similarity results of six typical attribute types in the XML Schema. Values in this table are computed based on the equation 3.11.

In Table 3.1, if two elements have the same data type, their compatible value is 1.00. Otherwise, this value is assigned by equation 3.11.

### 3.2.3 Constraint similarity

Another factor that affects the semantic similarity between two elements is the cardinality (occurrence) constraint. It is declared as *minOccurs* and *maxOccurs* in the XML Schema document. The *minOccurs* and *maxOccurs* respectively define the minimum and maximum number of occurrence times of an element in XML instances.

We use  $CSim(d_1, d_2)$  to specify the constraint similarity between two elements  $d_1$  and  $d_2$ . Different to the constraint table proposed in [2], in which values are decided by human judgment,

Table 3.1: Data type compatibility table

	URI	lang	text	Ubyte	dec	int	Dtime	Name	Entity	ID	Token	Type
URI	1.00	0.43	0.50	0.28	0.38	0.31	0.38	0.43	0.36	0.36	0.38	0.38
lang	0.43	1.00	0.43	0.28	0.31	0.27	0.31	0.39	0.33	0.33	0.35	0.32
text	0.50	0.43	1.00	0.33	0.38	0.31	0.38	0.43	0.36	0.36	0.39	0.41
Ubyte	0.28	0.28	0.33	1.00	0.33	0.53	0.49	0.28	0.22	0.22	0.24	0.23
dec	0.38	0.31	0.38	0.33	1.00	0.56	0.54	0.31	0.24	0.24	0.27	0.28
int	0.31	0.27	0.31	0.53	0.56	1.00	0.47	0.27	0.21	0.21	0.23	0.21
Dtime	0.38	0.31	0.38	0.49	0.54	0.47	1.00	0.31	0.24	0.24	0.27	0.28
Name	0.43	0.39	0.43	0.28	0.31	0.27	0.31	1.00	0.33	0.33	0.35	0.33
Entity	0.36	0.33	0.36	0.22	0.24	0.21	0.24	0.33	1.00	0.32	0.33	0.31
ID	0.36	0.33	0.36	0.22	0.24	0.21	0.24	0.33	0.32	1.00	0.33	0.31
Token	0.38	0.35	0.39	0.24	0.27	0.23	0.27	0.35	0.33	0.33	1.00	0.33
Type	0.38	0.32	0.41	0.23	0.28	0.21	0.28	0.33	0.31	0.31	0.33	1.00

Table 3.2: Cardinality constraint similarity table

	i=0, a = $\infty$	i=1, a = $\infty$	i=0, a=1	i=1, a=1
i=0, a = $\infty$	1.00	0.50	0.67	0.17
i=1, a = $\infty$	0.50	1.00	0.17	0.67
i=0, a=1	0.67	0.17	1.00	0.50
i=1, a=1	0.17	0.67	0.50	1.00

we define a novel metric to compute the constraint similarity values. For the definitely values of *minOccurs* and *maxOccurs*, we use the following equation for computing their cardinality constraint similarity:

$$CSim(e_1(min, max), e_2(min, max)) = \frac{(1 - \frac{|e_1.min - e_2.min|}{e_1.min + e_2.min}) + (1 - \frac{|e_1.max - e_2.max|}{e_1.max + e_2.max})}{2} \quad (3.12)$$

In equation 3.12, *min* and *max* are short forms of *minOccurs* and *maxOccurs*, respectively. Usually, *minOccurs* is assigned by 0 or 1, and *maxOccurs* is 1 or *unbound*. The value of *maxOccurs* is often undetermined (*unbound*). To measure the similarity for this value, we use the following function:

$$d_1[maxOccurs = \infty] = 4294967296 * MAX(d_2[maxOccurs]) \quad (3.13)$$

where 4294967296 is the maximum value declared for *maxOccurs* property, suggested by Microsoft [63]. We decide to use this function, since we have surveyed in our dataset (XSD and XML instances), the appearance time of an attribute with *maxOccurs=unbound* is about five times greater than maximum values of definitely *maxOccurs*. In the case that all *maxOccurs* in the XSD document are undetermined, we assign their values are 5. Taking this value, and then apply for the equation 3.12, we harvest the similarity of the attribute's cardinality constraint. The results are presented in Table 3.2.

Table 3.2 presents the cardinality constraint similarity with the value of unbound is 5. The characters  $i$ ,  $a$ , and  $\infty$  are the short forms of the *minOccurs*, *maxOccurs*, and *unbound*, respectively.

Sometimes, the values of the *minOccurs* and *maxOccurs* are retrieved from other indicators. For instance, *Order* and *Group* indicators (*any*, *all*, *choice*, *sequence*, *group name*, and *group reference*) have the default value for *maxOccurs* and *minOccurs* to be 1.

### 3.3 Structural Similarity Measurement (StSim)

The second stage is called *structure* similarity measurement. This stage measures the schema elements based on the similarity of their context (position) and their nearest elements. For instance, *housenumber* in Figure 3.3 is mapped to *postcode* in Figure 3.2, since they have the same parent *address* and the other three children (*street*, *city*, *country*) are also matched to each others.

The structure matching depends in part on the semantic similarity that is computed in the first stage. For instance, *given* and *family* in Figure 3.3 should match to *firstname* and *lastname* in Figure 3.2, rather than to the *type* and *authority* under element *id*, since element *id* in Figure 3.2 already matches to the same element in Figure 3.3. The result is a structure similarity coefficient, *StSim*, for each pair of elements.

The context of an element is composed of ancestor, sibling, immediate children and leaves. Two elements have a structural similarity if they are similar in contexts. In our algorithm, the structure similarity is computed based on the following principles:

- Elements that are leaves of the two trees are similar if their tags are similar, and the elements in their ancestors and siblings are similar.

- Two non-leaf elements are similar if their tags are similar, and the sub-tree rooted at the two elements are similar; and
- Two non-leaf elements are structurally similar if their leaf sets are highly similar, even if their immediate children are not.

According to the definition of *structure* in Section 2.1.4, we infer that in order to compute the structural similarity, we should have to measure the similarity of elements having functional connected with computed elements. Therefore, the structural similarity is drawn based on the following Definition 3.3:

**Definition 3.3:** The structural similarity (*StSim*) between two elements  $e_1$  and  $e_2$  is defined as weighted sum of ancestor similarity (*AcSim*), sibling similarity (*SbSim*), and children similarity (*ChSim*):

$$StSim(e_1, e_2) = \frac{\varepsilon_1 * AcSim(e_1, e_2) + \varepsilon_2 * SbSim(e_1, e_2) + \varepsilon_3 * ChSim(e_1, e_2)}{\varepsilon_1 + \varepsilon_2 + \varepsilon_3} \quad (3.14)$$

where  $\varepsilon_1$ ,  $\varepsilon_2$ , and  $\varepsilon_3$  are weighted parameters between 0 and 1. Details of each similarity factor in Equation 3.14 are presented in next subsections.

### 3.3.1 Ancestor similarity

Ancestor elements are the collection of elements from the current element to the root node. Exception for root element, each element in XML Schema has at least one ancestor. Therefore, the ancestor comparison is turned to the similarity computation of each ancestor of the first element with another ancestor of the second element. Assume that  $E_1$  and  $E_2$  are the collections of ancestor elements of  $e_1$  and  $e_2$ , respectively, then the ancestor similarity of these two elements is

presented as in matrices 3.15 and 3.16:

$$AcSim(E_1, E_2) = \begin{bmatrix} SeSim(e_{1_1}, e_{2_1}) & \cdots & SeSim(e_{1_1}, e_{2_n}) \\ \vdots & \ddots & \vdots \\ SeSim(e_{1_m}, e_{2_1}) & \cdots & SeSim(e_{1_m}, e_{2_n}) \end{bmatrix}, m \geq n \quad (3.15)$$

$$AcSim(E_2, E_1) = \begin{bmatrix} SeSim(e_{2_1}, e_{1_1}) & \cdots & SeSim(e_{2_1}, e_{1_m}) \\ \vdots & \ddots & \vdots \\ SeSim(e_{2_n}, e_{1_1}) & \cdots & SeSim(e_{2_n}, e_{1_m}) \end{bmatrix}, m < n \quad (3.16)$$

where  $m$  and  $n$  are the number of ancestor elements in the sets  $E_1$  and  $E_2$ , respectively;  $SeSim(e_1, e_2)$  is the semantic similarity between elements  $e_1$  and  $e_2$ , determined by equation 3.2. The ancestor similarities of two set elements  $E_1$  and  $E_2$  in the matrices 3.15 and 3.16 are determined by either following equation 3.17 or 3.18, respectively:

$$AcSim(E_1, E_2) = \frac{\sum_{i=1}^m \max_{j=1}^n (SeSim(e_{1_i}, e_{2_j}))}{m}, m \geq n \quad (3.17)$$

$$AcSim(E_2, E_1) = \frac{\sum_{i=1}^n \max_{j=1}^m (SeSim(e_{2_i}, e_{1_j}))}{n}, m < n \quad (3.18)$$

where  $max$  is the maximum similarity value of each row in the matrices. If two elements  $e_1$  and  $e_2$  do not have any ancestor element (it means they are root elements), then  $AcSim(e_1, e_2) = 1$ .

### 3.3.2 Sibling similarity

Similar to the computation of ancestor similarity, to compute the sibling similarity of between elements, we pick up a sibling element from the first element to compare with a sibling of the second element. For instance, the sibling of an element  $e_1$  is  $Se_1 = [e_{1_1}, e_{1_2}, \dots, e_{1_k}]$ , and the sibling of an element  $e_2$  is  $Se_2 = [e_{2_1}, e_{2_2}, \dots, e_{2_t}]$ , where  $k$  and  $t$  are the numbers of siblings of the element

$e_1$  and  $e_2$ , respectively. If  $k \geq t$ , we take each element in  $Se_1$  to compare with each element in  $Se_2$ . The highest value of the measurement is chosen. If  $k < t$ , we compare each element in  $Se_2$  with each element in  $Se_1$ . The sibling similarity of two elements is the average of the similarities of all sibling pairs. The sibling similarity is computed based on the following Equation 3.19 or 3.20:

$$SbSim(e_1, e_2) = \frac{\sum_{i=1}^k \max_{j=1}^t (SeSim(e_{1_i}, e_{2_j}))}{k}, k \geq t \quad (3.19)$$

$$SbSim(e_2, e_1) = \frac{\sum_{i=1}^t \max_{j=1}^k (SeSim(e_{2_i}, e_{1_j}))}{t}, k < t \quad (3.20)$$

where  $SeSim$  is the semantic similarity from Equation 3.2. In the case that both elements  $e_1$  and  $e_2$  do not have any sibling, the  $SbSim(e_2, e_1) = 1$ .

### 3.3.3 Children similarity

We say that a child element of the first compared element  $e_1$  has a link to another child of the second element  $e_2$  if their name similarity,  $NSim$ , exceeds the maximum threshold,  $thresh\_max$ . We choose the low value of  $thresh\_max$  in order to increase the value of children similarity measurement. The children similarity is determined by following Equation 3.21.

$$SbSim(e_1, e_2) = \frac{nlinks(e_1, e_2) + nlinks(e_2, e_1)}{leaves(e_1) + leaves(e_2)} \quad (3.21)$$

where  $leaves(e_1)$  is the total number of children (leaves) of the element  $e_1$ ;  $nlinks(e_1, e_2)$  is the total number of links from the children of element  $e_1$  to children of the element  $e_2$ .

For example, let's compute the children similarity of two elements, *address* and *address'* in Figure 3.2 and Figure 3.3, respectively:

$$leaves(address) + leaves(address') = 4 + 4 = 8$$

$$nlink(address, address') = 3$$

$nlink(address', address)=3$

$$\implies SbSim(address, address')=\frac{6}{8}=0.75$$

If one of the compared elements is a leaf node (contains no child nodes), their children similarity is 0, otherwise, if both compared elements are leaf nodes, the  $SbSim(e_1, e_2) = 1$ .

### 3.4 Similarity between Two Schema Trees

The structure similarity measurement of two schema trees is presented in Figure 3.6. In Figure 3.6, we choose the minimum value of threshold is 0, and maximum is 0.3. Every time of comparing one of two values is changed by 0.1. The maximum threshold can be altered. Because in the first stage of computing semantic similarity, we select the maximum element depth 3, so it is same with structural measurement.

In contrary to the semantic computation, the depth-first search order, the structural relatedness computation visits the element from the leaf node to the root node, following post-order traversal algorithm.

The elements in the two trees are then enumerated in post-order, which is uniquely defined for a given tree. The first step in the loop calculates the structure similarity between two elements. For leaves, this is just the value of  $StSim$  that is calculated in the previous step. When one of the two elements is not a leaf, the structural similarity is calculated as a measure of the number of leaf level matches in the sub-trees.

In order to compute the similarity between two schema trees, we have to complete the structure similarity between each pair of elements in two trees. For each pair of schema elements, the algorithm compares the structural similarity,  $StSim$ . It is the similarity of the contexts in which the elements occur in the two schemas.



---

```

1  Algorithm: Tree_similarity;
2      //It is based on the structural similarity, StSim
3  Input: Two schemas S and T
4  Output: The structural similarity
5  Begin
6  thresh_min=0; thresh_max=0.3;
7  for each s in S, t in T where s, t are leaves
8      S'=post-order(S);
9      T'=post-order(T);
10 for each s in S'
11     for each t in T'
12         if StSim(st) >= thresh_max then
13             StSim(s,t) = StSim(s,t)+0.1;
14         else if StSim(s,t) <= thresh_min then
15             StSim(s,t) = StSim(s,t) - 0.1;
16 Tree_similarity(S,T) = StSim(s,t)
17 End;

```

---

Figure 3.6: The structure similarity algorithm

To understand our algorithm, let us compute the element similarity of some pairs of elements given in Figure 3.2 and Figure 3.3. To distinguish between elements with the name labels in two schemas, we put an apostrophe in the element name of the second schema.

For example, we compute the element similarity of the elements *person.name* and *name* in Figure 3.2 and Figure 3.3, respectively. Because two elements have differences in their names, we have to compute their linguistic similarity by using *WordNet* [94]. The element *person.name* is tokenized as *person* and *name*. The last token is matched with an element *name* in Figure 3.2.

Therefore, the linguistic similarity of *person.name* and *name* is 0.8. Because both elements are complex types, their data type similarity is 0. Moreover, their cardinality constraint value is the same, so the constraint similarity is 1.

When we have the element similarity of all element pairs in two schemas, we can compute the similarity of the two schemas. The similarity between two XML Schemas trees is calculated as the weighted sum of two components:

$$SSim(T_1, T_2) = \varepsilon * \sum_{i=1}^k SeSim(T_1, T_2) + (1 - \varepsilon) * TreeSim(T_1, T_2) \quad (3.22)$$

where *SSim* is the schema tree similarity;  $\varepsilon$  is the weighted value, between 0 and 1;  $k$  is the lowest number of elements in the tree, for example, if the schema  $T_1$  has total 230 elements, schema  $T_2$  has 220 elements, then  $k = 220$ ;  $e_1$  and  $e_2$  are the elements of  $T_1$  and  $T_2$ , respectively; *TreeSim* is the tree similarity of two schemas.

The similarity calculation has a close relationship to each other and a recursion. Two elements are semantic similar if their leaf sets are similar. The semantic similarity of the leaves is increases if their ancestors are highly similar. The similarity of the structure is also influenced by the semantic similarity. If the sub-tree of two elements are high similar, the structure similarity of their ancestor is high, too.

### 3.5 XML Schema Integration

In this section we shall describe the process of XML Schema integration based on the above similarity measure. Figure 3.7 shows our clustering framework, which consists of two main phases: *Similarity computation* (ESim) and *Clustering*. The similarity computation phase composes of semantic similarity and structural similarity which are described in Sections 3.2 and 3.3, respectively. In this phase the similarity of all XML schema pairs are assessed to form the similarity

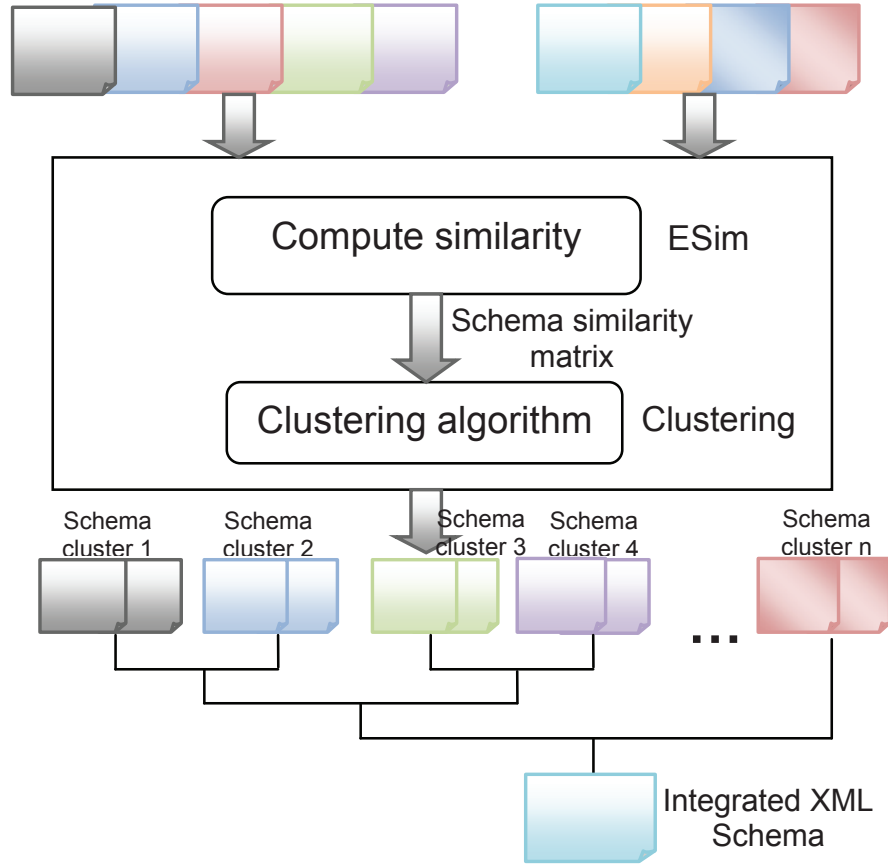


Figure 3.7: XML Schema integration framework architecture

matrix, which will then be used by the clustering algorithm.

Given a set of XML Schemas  $S = XSD_1, XSD_2, \dots, XSD_n$ , we construct a  $n \times n$  schema similarity matrix  $ESimMat$ . Each entry in the matrix  $ESimMat[i][j]$  represents the similarity between schema  $XSD_i$  and  $XSD_j$ . For every schema pair, we sum up (weighted sum) all element similarity values computed by the semantic similarity and structural similarity.

Clustering of XSD can be carried out once we have the XSD similarity matrix. There are

many techniques for clustering algorithms among them are hierarchical clustering [58]. Hierarchical clustering solutions are in the form of trees called *dendrograms*, which provide a view of the data at different levels of abstraction. XSDs from the same application domain tend to be clustered together and form clusters at different cut-off values. The consistency of clustering solutions at different levels of granularity allows flat partitions of different granularity to be extracted during data analysis, making them ideal for interactive exploration and visualization [104].

However, *quantifying* the *goodness* of an integrated schema remains an open problem since the integration process is often subjective. One integrator may decide to take union of all elements in the XSD documents, while another may prefer to retain only the common XSD elements in the integrated schema. Here, we adopt the union approach which avoids loss of information. In addition, the integrated XSD should be as compact as possible. In other words, we define the quality of an integrated schema as inversely proportional to its size, that is, a more compact integrated XSD is the result of a better integration process.

## Chapter 4

# S-Trans: Duplicate Similarity Measure for XML2OWL

This chapter presents the general mapping modules which are necessary for transformation of XML data into OWL ontology, the semantic similarity measurement of XML duplicates, and the transformation of XML schema (XSD or DTD) into OWL ontology (XML2OWL).

### 4.1 General modules of XML2OWL Transformation

In general, in order to harvest data in the form of the ontology language, in this case OWL, the following modules are required. The first area is *extracting manager*. It draws the necessary data file from different data sources. The second thing is the *mapping modules* that map the data source to ontology model. Other important modules are the *query handler*, which draws information from data sources, the *instance generator*, which transforms the input instances into the output individuals, and lastly the *ontology schema*, that provides a machine readable format (in our case is OWL model) for improving the semantics of data sources. Figure 4.1 represents the high level illustration of **S2S** (syntactic to semantic) architecture [85].

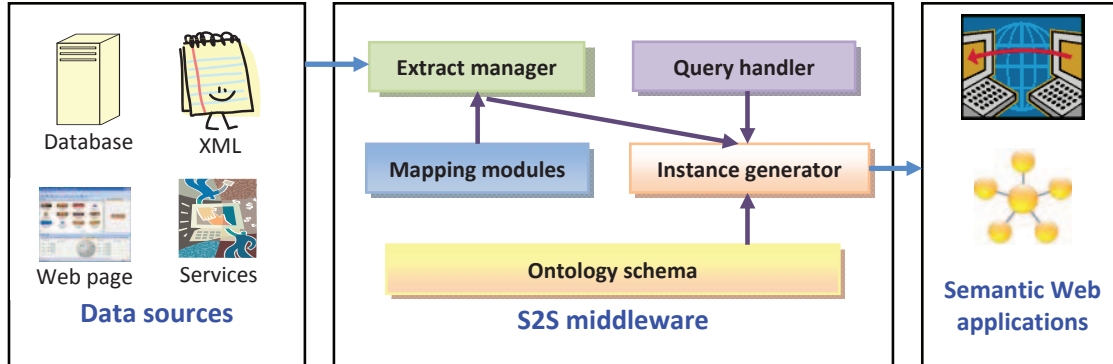


Figure 4.1: General syntactic to semantic architecture

Since our approach has already chosen XML and its schema (DTD or XSD) as the data source, we need not the *extract manager*. Moreover, we aim at converting all the given information (not some desired facts) in the XML document to OWL format by traversing from the beginning to the end of the document, so the *query handler* is not used. We principally concentrate on the *transforming from DTD or XSD into OWL ontology* and *instance generating* from XML document to OWL instances.

Although some approaches have been developed to transform XSD or DTD into the OWL ontology, several problems must be resolved. One is the problem of duplicate elements in an XSD or DTD document. Most transforming approaches provide a unique identifier for each schema (XSD or DTD) element by adding a new key element or changing the source element's name [26], [4], [3], [10], [92], [3], [82]. However, this solution may lead to data redundancy because duplicate elements may represent the same information. The perfect XML transformation should create a correct, complete, and unique representation of every concept. To obtain this data quality, a similarity computation of duplicate elements is used. In this computation, if two elements have highly similar semantics, they are transformed into one representation.

## 4.2 Semantic Similarity of Duplicate Elements

### 4.2.1 Motivating example

To illustrate the **S-Trans** method, we will first restrict ourselves to the hierarchical schemas. The XML schema is displayed as a graph, where the nodes represent the schema elements. We motivate **S-Trans** with the real XML data set, *prescription.dtd* [20]. The DTD document and its corresponding XSD are displayed in Figures 4.2 and 4.3.

Assuming that all forms of XML data, such as XSD, DTD, and XML instances, can be represented as a tree, we draw a tree of *prescription.dtd* as in Figure 4.4. As presented in Figure 4.4, the element *name* of the *physician* is similar to the element *name* of the *patient* and is different from the element *name* of the *drug* because *name* of the *drug* contains two children, and the siblings of *name* of the *drug* are slightly different from those of *name* of the *physician* and *name* of the *patient* elements.

In contrast, two duplicate elements, *phone* of the *physician* and *phone* of the *patient*, do not contain any children, but they are different in their cardinality constraint. The first *phone* element is declared with '+', whereas the second one is defined with '\*'. Furthermore, *month*, *date*, and *year* of *prescribed.date* are repeated in the *DOB* (Date Of Birth) element. These elements have the same cardinality constraint, the same siblings, and children, but they have different parents.

On the basis of the above mentioned observations, we can conclude that the similarity between duplicate elements is affected by both internal factors (such as name, data type, and cardinality constraint) and external factors (ancestor, sibling, and children). Therefore, our duplicate similarity measure is the combination of these factors using a weighted function, similar to Equation 3.1 in previous chapter, but in this case we only measure the similarity of duplicates within an XML schema (XSD or DTD). Our duplicate similarity is determined as following Definition 4.1:

---

```
1
2 <!ELEMENT prescription (prescribed.date,
3     patient, drug, physician, interchange?)>
4 <!ELEMENT prescribed.date (month, day, year)>
5 <!ELEMENT patient (name, id*, gender, DOB, address+, phone*)>
6 <!ELEMENT drug (drug.name, strength, quantity, sig)>
7 <!ELEMENT name (#PCDATA)>
8 <!ATLIST name liquid NMTOKEN #IMPLIED>
9 <!ATLIST name tablet NMTOKEN #IMPLIED>
10 <!ELEMENT id (SSN|DEA)>
11 <!ELEMENT SSN (#PCDATA)>
12 <!ELEMENT DEA (#PCDATA)>
13 <!ELEMENT gender (#PCDATA)>
14 <!ELEMENT DOB (month, day, year)>
15 <!ELEMENT address (#PCDATA)>
16 <!ELEMENT phone (#PCDATA)>
17 <!ELEMENT physician (name, address+, phone+, id+)>
18 <!ELEMENT month (#PCDATA)>
19 <!ELEMENT day (#PCDATA)>
20 <!ELEMENT year (#PCDATA)>
21 <!ELEMENT interchange (#PCDATA)>
```

---

Figure 4.2: Example of a DTD document, prescription.dtd



---

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3   <xs:element name="prescription">
4     <xs:complexType> <xs:sequence>
5       <xs:element ref="prescribed.date"/>
6       <xs:element ref="patient"/>
7       <xs:element ref="drug"/>
8       <xs:element ref="physician"/>
9       <xs:element ref="interchange" minOccurs="0" maxOccurs="1"/>
10    </xs:sequence> </xs:complexType>
11  </xs:element>
12  <xs:element name="patient">
13    <xs:complexType> <xs:sequence>
14      <xs:element ref="name"/>
15      <xs:element ref="id" minOccurs="1" maxOccurs="unbounded"/>
16      <xs:element ref="gender"/>
17      <xs:element ref="DOB"/>
18      <xs:element ref="address" minOccurs="1" maxOccurs="unbounded"/>
19      <xs:element ref="phone" minOccurs="0" maxOccurs="unbounded"/>
20    </xs:sequence> </xs:complexType>
21  </xs:element>
22  <xs:element name="name">
23    <xs:complexType mixed="true">
24      <xs:attribute name="liquid" type="xs:NMTOKEN" use="optional"/>
25      <xs:attribute name="tablet" type="xs:NMTOKEN" use="optional"/>
26    </xs:complexType> </xs:element>
27    ...
28 </xs:schema>

```

---

Figure 4.3: Example of a DTD and a part of its corresponding XSD document

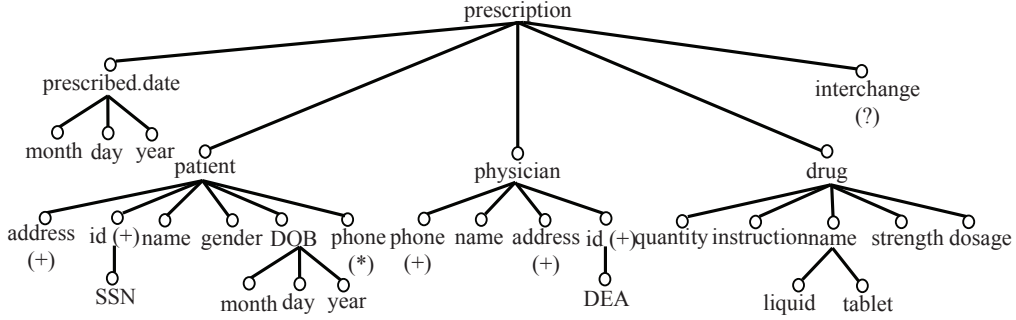


Figure 4.4: The corresponding tree of XML schema (XSD/DTD) in Figure 4.2

**Definition 4.1.** The duplicate similarity,  $DupSim$ , between duplicate elements,  $e_1$  and  $e_2$ , in XSD or DTD document is defined as the weighted sum of the semantic similarity ( $SeSim$ ) and the structural similarity ( $StrSim$ ):

$$DupSim(e_1, e_2) = \alpha * SeSim(e_1, e_2) + (1 - \alpha) * StrSim(e_1, e_2) \quad (4.1)$$

where  $\alpha_i$  is the weight parameter to balance the role of  $SeSim$  and  $StrSim$ . Similar to Equation 3.1,  $\alpha_i = 0.55$  according to our experiment.

Reminding that  $DupSim$  in this chapter measures the similarity of duplicates within a single schema, whereas the  $ESim$  in the previous chapter measures the similarity of elements between different schemas. Therefore, we can use the semantic similarity measure ( $SeSim$ ) in Equation 3.2 to compute the  $SeSim$  similarity of duplicate elements. However, the structural similarity of  $DupSim$  is different from  $ESim$ . The structural similarity of  $DupSim$  is determined by the following Definition 4.2:

**Definition 4.2.** The structural similarity,  $StrSim$ , between duplicate elements,  $e_1$  and  $e_2$ , in

XSD or DTD document is defined as the weighted sum of the ancestor similarity (*ASim*), the sibling similarity *SbSim*, and the descendant similarity (*DeSim*):

$$StrSim(e_1, e_2) = \eta * ASim(e_1, e_2) + \theta * SbSim(e_1, e_2) + (1 - \eta - \theta) * DeSim(e_1, e_2) \quad (4.2)$$

where  $\eta$  and  $\theta$  are the weighted parameters. Since the roles of *ASim* and *SbSim* are assumed to be equivalent, we assign 0.33 to  $\eta$  and  $\theta$ . Details of each measuring factor are presented in next subsections.

#### 4.2.2 Ancestor similarity (*ASim*)

In order to measure the ancestor similarity of duplicate elements, the important thing is to find their common super concept. Starting from two duplicate elements, the algorithm traverses each of their ancestor elements in turn until it finds their common node. Suppose that *C* is the least common super concept of the duplicate elements  $e_1$  and  $e_2$ , and  $level_{e_1}$  and  $level_{e_2}$  are the numbers of nodes on the path from  $e_1$  and  $e_2$ , respectively, to *C*. Then their ancestor similarity is determined by following equation:

$$ASim = \begin{cases} 1 & \text{if } level_{e_1} = level_{e_2} = 1 \\ 0.85^{max(m,n)-1} & \text{otherwise} \end{cases}$$

The value of 0.85 is chosen based on our observation in Figure 4.5, which presents the influence of five candidate values (0.9, 0.85, 0.8, 0.75, and 0.7) on the *ASim*. Since we expect the ancestor similarity to be greater than 0.7 when the maximum number of ancestor nodes is 3 ( $level = 2$ ), we choose the appropriate value that satisfies this requirement. As can be seen in Figure 4.5 at level 2, the ancestor similarities of *ASim*.0.85 and *ASim*.0.9 are 0.72 and 0.81, respectively, so we select 0.85 to be the power function for computing the ancestor similarity.

In order to compare the similarity of the ancestor nodes, the semantic similarity (*SeSim*) is used. If their *SeSim* values are the same, then  $ASim = 1$ ; otherwise, we compare their grandparent:

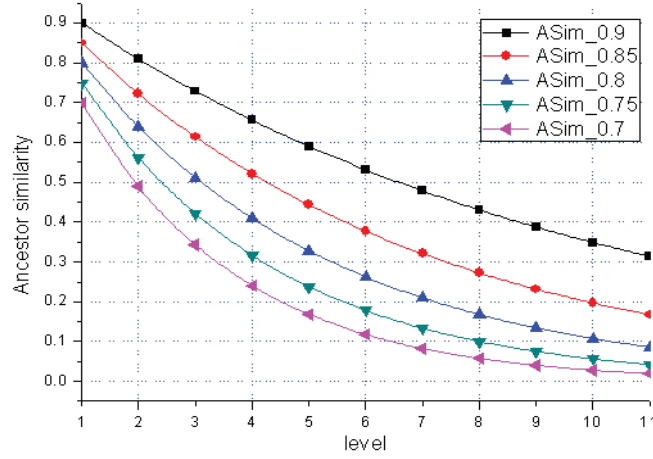


Figure 4.5: The ancestor similarities at different ancestor levels with five candidate values

the first grandparent has level 1, the second grandparent has level 2, and so on. The value of  $ASim$  is the exponent function  $0.85^{level}$ . Details of the parent similarity algorithm for duplicate elements are presented in Figure 4.6.

In Figure 4.6, we assume the maximum distance of the twin elements to their common super concepts is 11. This is because in Figure 4.5 at  $level=11$ , the  $ASim$  value of  $ASim_{0.85}$  is very small (approximately 0.17), so our ancestor algorithm does not compute for that level and higher.

### 4.2.3 Sibling similarity (SbSim)

Similar to the sibling measuring metrics in Section 3.3.2, the sibling similarity between duplicates is computed in the same way with the sibling similarity between elements in different documents in Equations 3.19 and 3.20.

```

Input: Two elements with the same name but in XSD tree,  $e_1$  and  $e_2$ 
Output: The ancestor similarity
    level== 0;max_level==11;
Function ASim( $e_1, e_2, level$ )
    if ((SeSim( $e_1, e_2$ )==1)) or (SeSim(parent:: $e_1$ ,parent:: $e_2$ )==1)
        then return 1;
    else if ((SeSim( $e_1$ ,parent:: $e_2$ )==1) or (SeSim(parent:: $e_1$ ,parent:: $e_2$ )==1)
        then return 0.85;
    else if (level == max_level) return 0;
    else
        return
        power(0.85*ASim(parent:: $e_1$ , parent:: $e_2$ ,level+1));
    end;
End;

```

Figure 4.6: The ancestor similarity algorithm

#### 4.2.4 Children similarity (ChSim)

Different from children similarity in ESim method, the children similarity of duplicate elements computes not only the similarity of direct children but also all descendant elements of both duplicates. To compute the children similarity of duplicates, first of all, the algorithm collects all descendant elements of each duplicate. For instance, the children of the first duplicate  $e_1$  is  $E_1$ , where  $E_1 = [e_{1_1}, e_{1_2}, \dots, e_{1_r}]$ , and the sibling of a duplicate  $e_2$  is  $E_2$ , where  $E_2 = [e_{2_1}, e_{2_2}, \dots, e_{2_s}]$ , where  $r$  and  $s$  are the total number of descendant elements of duplicates  $e_1$  and  $e_2$ , respectively.

After having two sets of children elements, the algorithm continues to compute the semantic similarity of each child in the  $E_1$  with another in  $E_2$ . There are two cases for comparison: If  $r \geq s$ , the algorithm takes each element in  $E_1$  to compare with each element in  $E_2$ ; otherwise the algorithm picks each element in  $E_2$  to compare with an element in  $E_1$ . Those comparison can be described by following matrices 4.3 and 4.4:

$$ChSim(e_1, e_2) = \begin{bmatrix} SeSim(e_{1_1}, e_{2_1}) & \cdots & SeSim(e_{1_1}, e_{2_s}) \\ \vdots & \ddots & \vdots \\ SeSim(e_{1_r}, e_{2_1}) & \cdots & SeSim(e_{1_r}, e_{2_s}) \end{bmatrix}, r \geq s \quad (4.3)$$

$$ChSim(e_2, e_1) = \begin{bmatrix} SeSim(e_{2_1}, e_{1_1}) & \cdots & SeSim(e_{2_1}, e_{1_r}) \\ \vdots & \ddots & \vdots \\ SeSim(e_{2_s}, e_{1_1}) & \cdots & SeSim(e_{2_s}, e_{1_r}) \end{bmatrix}, r < s \quad (4.4)$$

where  $SeSim(e_1, e_2)$  is the semantic similarity of elements  $e_1$  and  $e_2$ , which is determined by Equation 3.2. The children similarities of two duplicates  $e_1$  and  $e_2$  in the matrices 4.3 and 4.4 are determined by Equations 4.5 and 4.6, respectively:

$$ChSim(e_1, e_2) = \frac{\sum_{i=1}^r \max_{j=1}^s (SeSim(e_{1_i}, e_{2_j}))}{r}, r \geq s \quad (4.5)$$

$$ChSim(e_2, e_1) = \frac{\sum_{i=1}^s \max_{j=1}^r (SeSim(e_{2_i}, e_{1_j}))}{s}, r < s \quad (4.6)$$

In the case that either element  $e_1$  or  $e_2$  is a leaf node (meaning it contains no child node), the child similarity of this twin element is 0. If both are leaf nodes, their children similarity  $ChSim = 1$ .

Depending on the expected similarity value, the duplicate elements can be classified into two groups, similar and non-similar. The transforming strategies in Section 4.3 are then applied to transform these duplicates into the appropriate OWL concepts. In this paper, we use the threshold value 0.7 to classify the duplicate elements. The value of 0.7 was chosen based on our observation of experimental results: At the threshold of 0.7, the error rate of classification is greatly smaller than other thresholds. **See Section 5.2 for details.**

### 4.3 Transforming DTD/XSD into the OWL Ontology

The transformation framework of XML into OWL is shown in Figure 4.7. There are four main steps in the framework: Extracting schema from the XML instances; Measuring duplicate similarity in XML schema; Transforming elements in XML schema into OWL concepts; and Transforming XML instances into OWL individual. First, we extract XSD or DTD schema. This step only works when an XML document does not go with a schema. We generate an XSD or DTD schema corresponding to this XML document by using the available tool recommended by HIT software [6].

Second, we compute the similarity of duplicates within an XML schema. This step is necessary to eliminate the redundancy and improve the quality of OWL results. The details of duplicate computation is presented in Section 4.2. Third, a schema transformation step will transform all of XSD elements into OWL ontology which captures the semantics and maintains the XML structure. Once OWL concepts are created, S-Trans system moves to the fourth step. The input of this step is an XML document together with an OWL ontology generated from the previous step. The S-Trans traverses from the root element in the XML document and ends when it meets a close-tag of the root element. If an element in XML data is matched with a node in an OWL ontology, XML2OWL procedure will execute the transformation and generate an OWL individual document.

According to the characteristics of OWL definitions and the functionality of DTD elements, we define suitable transforming notations from DTD elements into OWL concepts. In OWL, a class defined by *owl:class* identifies a class or a non-instance item of the ontology. Because *DOCTYPE* defining the root element of the document usually contains other elements and attributes, it is converted to an *owl:class*. If *ELEMENT* includes other elements or attributes or refers to an entity notion, it is mapped to the *owl:class* as well. *ATTLIST*, which normally defines the attributes

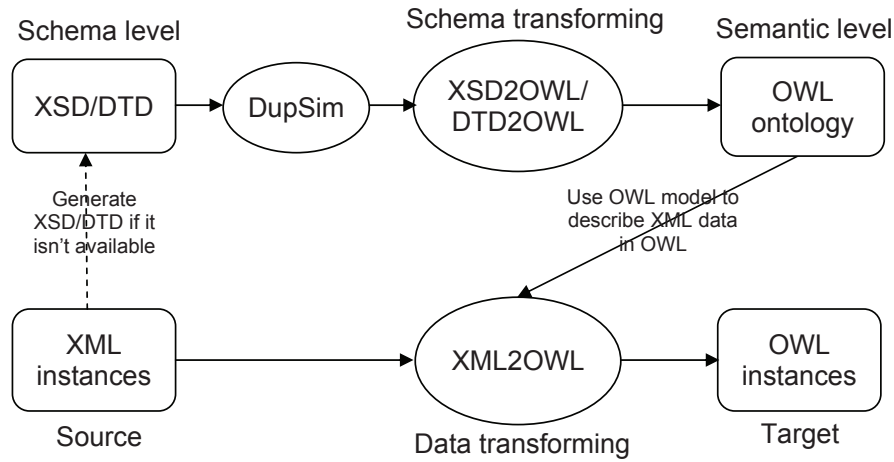


Figure 4.7: Transforming framework from XML into OWL

of the document, is transformed into *owl:DatatypeProperty* by default. However, if *ATTLIST* contains other entity references, we map it to an *owl:class* because it has the same functionality of the class. This notion was not mentioned in our previous work [91].

In OWL, a property is divided into two types, *ObjectProperty* and *DatatypeProperty*. Because *ObjectProperty* specifies the relationship between two instances that belong to the same or different classes [10], we use *ObjectProperty* to describe the relationship among OWL classes. The *DatatypeProperty* indicates the relationship between instances and RDF literals. The *rdfs:domain* and *rdfs:range*, which restrict the anterior and posterior values of a property, respectively, are used as a supplement for the *DatatypeProperty* and *ObjectProperty*.

Moreover, to prevent a member of one class from being a member of another class, we used *owl:disjointWith*. Other OWL descriptions such as *owl:unionOf*, *owl:DataRange*, *owl:oneOf*, *rdf:first*, *rdf:rest*, etc. are also utilized to improve the expressiveness of OWL attributes. Details of the transformation model from DTD and XSD into OWL are presented in Figure 4.8.





DTD		XSD	OWL representation		
			Type	rdfs:domain	rdfs:range
DOCTYPE (root)		element@name, complexType	owl:class, owl:disjointWith owl:ObjectProperty	class name	child name
	other elements	element@name, complexType	owl:class, owl:disjointWith owl:ObjectProperty	class name	child name
ELEMENT contains	ENTITY reference only	element@ref	owl:class, owl:disjointWith		
	data type only	element@name, complexType mix="true" simpleTypeelement@type	owl:DatatypeProperty	attribute name	datatype
ENTITY contains	>1 attributes	element@name, complexType mix="true" simpleType >1 attribute@name	owl:DatatypeProperty, owl:subPropertyOf	attribute name	datatype
	one attribute	element@name, complexType mix="true" simpleTypeelement@type	owl:DatatypeProperty	attribute name	datatype
ATTLIST contains	other property	>1 attribute@name, extension@base  restriction@base	owl:DatatypeProperty rdfs:subPropertyOf	attribute name	datatype
	data type only	1 attribute name	owl:DatatypeProperty	attribute name	datatype
ELEMENT element-name (child1,child2,...) +, *, ?, ?	ENTITY reference	attribute@ref	owl:class, owl:ObjectProperty	class name	child name
	sequence		owl:intersectionOf		
	choice		owl:unionOf		
	maxOccurs   minOccurs		owl:maxCardinality, owl:minCardinality		

Figure 4.8: The transforming correspondences between DTD/XSD and OWL

Furthermore, unlike XML, OWL does not allow identical names. Thus, when our procedure meets an element or attribute that has the same name as the previous node, two solutions are proposed:

1. If this duplicate element is highly similar to the previous element, the procedure uses *owl:unionOf* to connect the parent nodes of these duplicates in the same domain.
2. Otherwise, the procedure renames the duplicated element by adding the parent element's name along with an underscore '\_' character between the parent's name and the duplicate's name.

For example, because the value of *DupSim* between elements *name* of the *patient* and *name* of the *physician* is 0.8, which is higher than our threshold, the two elements *name* are combined to one element *name* and their domain consists of the parents of the two elements. The OWL description of two elements *name* is as Figure 4.9 follows:

However, because *DupSim* value between elements *name* of the *drug* and *name* of the *physician* (or *patient*) is 0.48, which is lower than our threshold (0.7), the element *name* of the *drug* is renamed by adding its parent name before its name. Its OWL result is presented in Figure 4.10

---

```
1 <owl:DatatypeProperty rdf:about="#name">
2   <rdfs:domain>
3     <owl:Class>
4       <owl:unionOf rdf:parseType="Collection">
5         <owl:Class rdf:about="#physician"/>
6         <owl:Class rdf:about="#patient"/>
7       </owl:unionOf>
8     </owl:Class>
9   </rdfs:domain>
10  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
11 </owl:DatatypeProperty>
```

---

Figure 4.9: OWL results of duplicates which are highly similar

---

```
1 <owl:DatatypeProperty rdf:ID="drug_name">
2   <rdfs:domain rdf:resource="#drug"/>
3   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
4 </owl:DatatypeProperty>
```

---

Figure 4.10: OWL results of duplicates which are less similar

## Chapter 5

# Experimental Results

The experiments are conducted to qualify the XSDs similarity measure, *ESim*, and the duplicate similarity measure in XML transformation into OWL, *S-Trans*, which are presented in Chapter 3 and Chapter 4, respectively. We evaluate *ESim* and *S-Trans* by matching two XSD documents and matching between XSD and OWL ontology. The experiments of each technique contain three main steps:

1. **Experimental setup:** This step includes the choosing of real-world XSDs, collecting the mutual similarities of duplicates and similarities of element pairs between synthetic XSDs.
2. **Parameters determination:** This step produces the appropriate values for weight parameters in similarity measuring metrics and for classifying value of duplicates.
3. **Evaluation:** This step compares the proposed methods with related work and evaluate the role of each measuring factor.

## 5.1 Experiments on XML Schemas Similarity Measure

The experiments of ESim have two phases: preparatory and experimental. In the first phase we need to determine the weight values of parameters of the similarity measure so that it provides realistic results. In the second phase, we analyze the behavior of the proposed metrics on real-world XML Schemas.

### 5.1.1 Determining of parameter values

The experimental implementation enable one to set the following parameters:

- weight  $\alpha$  and the second weight in ESim equation - equation (3.1). However, only  $\alpha$  needs to be set, since the second weight is the inverse value of  $\alpha$ , (equal  $1-\alpha$ ).
- weights  $\beta$ ,  $\chi$ , and the third weight in SeSim equation - equation (3.2). In the same way, the third weight is determined by  $1 - (\beta + \chi)$ .

Most of the current paper claim that the setting of similarity parameters can be determined by user and, hence, it is not discussed. The problem is how to prepare a reasonable setting so that the similarity measure returns reasonable results. For this purpose, we use the following strategy: Firstly, we prepare a set of synthetic XSDs and we determine their mutual similarity from user's perspective. Then, we set the respective parameters so that the similarity measure returns similar results. We depict the strategy using four XSDs describing patients (Patient\_A in Figure 3.1, Patient\_B in Figure 3.2, Patient\_C in Figure 5.1, and Patient\_D in Figure 5.2). Their mutual user-specified similarity is listed in Table 5.1.

Firstly, we set the weight parameters of SeSim equation (3.2) as following:  $\beta = 0.34$ , two other weight parameters share the same value 0.33 (i.e. we use generally acknowledged "reasonable" values) and analyze the results of similarity evaluation with changing weight  $\alpha = 0.05, 0.1, 0.15$ ,

Table 5.1: The similarity of synthetic XSDs

	Patient_A	Patient_B	Patient_C	Patient_D
Patient_A	1	0.86	0.48	0.5
Patient_B	0.86	1	0.4	0.37
Patient_C	0.48	0.4	1	0.8
Patient_D	0.5	0.37	0.8	1

..., 1 and respective value of the second weight ( $1 - \alpha$ ). The results are presented in Figure 5.3.

In Figure 5.3, the X-axis represents the values of  $\alpha$  and the lines represent the similarity of respective pairs of XSDs. As we can see, the most reasonable values of  $\alpha$  corresponding to similarity results expected by a user are represented using the black dots and occur within the interval of  $[0.5, 0.65]$  stressed using the vertical lines. The only exception is the similarity of XSDs C and D, which demonstrates the classical situation that user-specified similarity is not precise. Based on this observation, we choose the value for  $\alpha$  is  $0.55$ , and therefore the second weight in ESim function is  $0.45$ .

Secondly, we find the weight values of SeSim equation. We perform the same experiment with  $\alpha = 0.55$ . Since there are three weight parameters in SeSim equation:  $\beta$ ,  $\chi$ , and the third weight. The weight  $\beta$  is associated with the name similarity which is a very important factor in determining the similarity between elements, whereas  $\chi$  is associated with data type similarity which has a similar role as constraint similarity. Therefore, we divide three parameters into two groups: first  $\beta$  group and the second  $(1-\beta)$  group. The values of  $\chi$  and the third parameter is the fraction of  $\frac{(1-\beta)}{2}$ . The results are depicted in Figure 5.4.

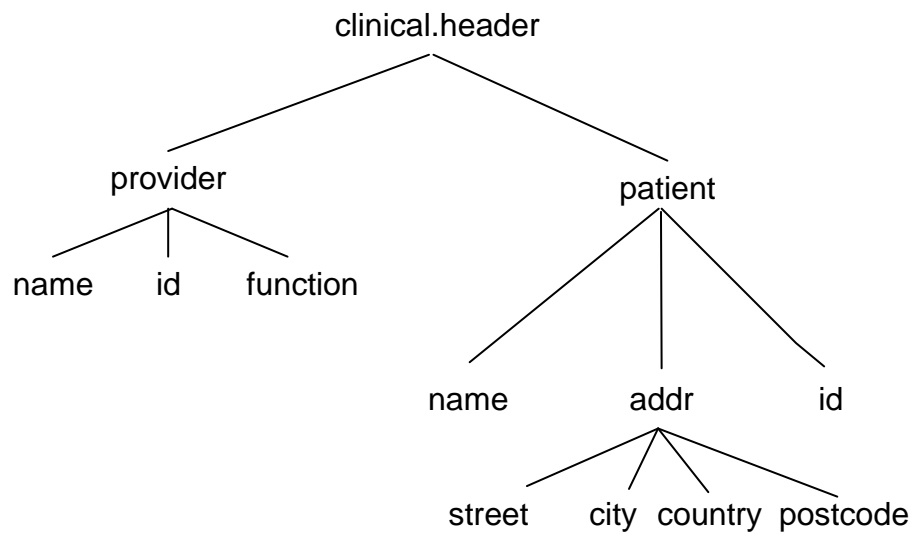


Figure 5.1: Tree representation for Schema Patient\_C

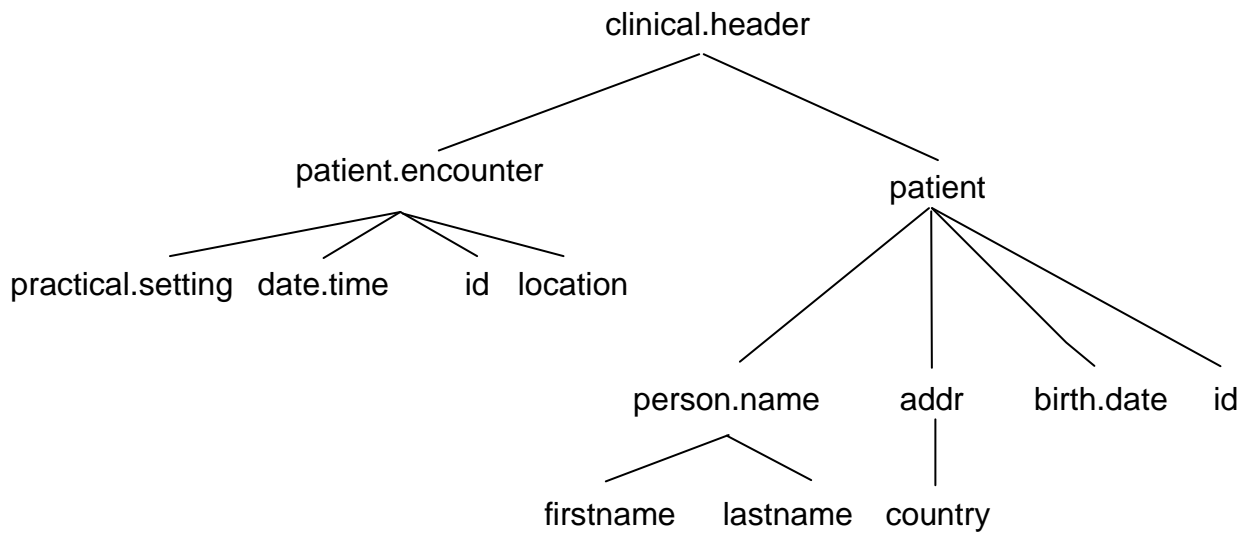


Figure 5.2: Tree representation for Schema Patient\_D



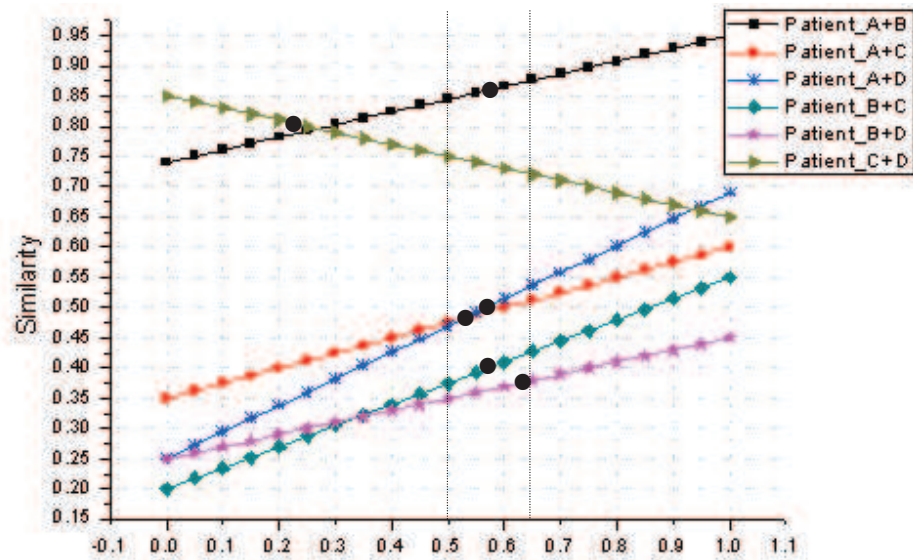


Figure 5.3: Determining weights of ESim function

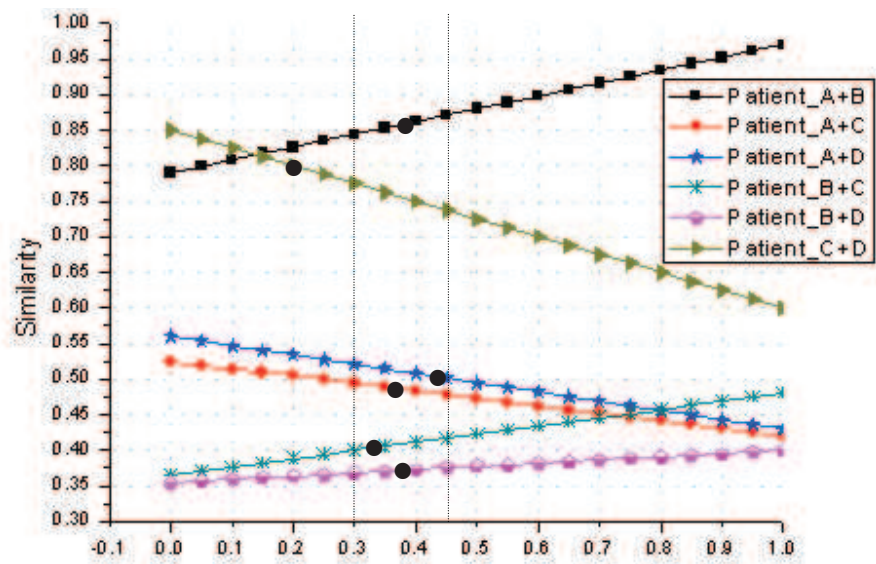


Figure 5.4: Determining weights of SeSim function

Table 5.2: The characteristics of the tested schemas

	Schema 1 vs Schema 2	#nodes	Average #nodes	#max depth
1	Patient_A vs Patient_B	12/10	11	3/3
2	Healthcaremetadata vs healthcarevocabulary	137/29	83	7/4
3	Yahoo Finance vs Standard	10/16	13	2/2
4	Cornell vs Washington	34/39	36	3/3
5	CIDX vs Excel	30/40	35	3/4
6	Google vs Looksmart	706/1081	893	11/16
7	Google vs Yahoo	561/665	613	11/11
8	Yahoo vs Looksmart	74/140	107	8/10
9	Iconclass vs Aria	999/553	776	9/3

As can we see in Figure 5.4, the interval where the expected similarity values (denoted again using black dots) occurs in  $[0.3, 0.45]$ . Note that in this case, the similarity between *Patient\_C* and *Patient\_D* is also diverse, even beyond the scope of the results of the algorithm. Therefore, we choose the weight value for parameter  $\beta$  is  $0.4$ , so the parameters  $\chi$  and the left one share the same value  $0.3$ . We use those values for experimentation with the real-world data sets.

### 5.1.2 Results based on real-world XSDs

The element similarity including semantic and structural similarity is implemented by C# language [62]. To examine the performance of *ESim*, we use healthcare XML Schemas (and corresponding DTDs) from [61], [21] and tested schemas from xCBL [84]. The experiment for *ESim* is executed on nine pairs of schemas, which are presented in Table 5.2.

In Table 5.2, the test case #1 comes from Figure 3.1 and Figure 3.2, the test case #2 is from Mebiquitous schema [61], and other test cases are from xCBL [84]. In general, there are two

similarity test cases on healthcare domain (#1 and #2), two test cases on business domain (#3, #5). The first business example (#3) describes two company profiles: a Standard one and Yahoo Finance. The second business example (#5) deals with BizTalk purchase order schemas. There is one measuring task from an academy domain (#4). It describes courses taught at Cornell University and at the University of Washington. There are three measuring tasks on general topics (#6, 7, 8) are represented by the well-known web directories, such as Google, Yahoo, and Looksmart. Finally, the last measuring task (#9) is from the cultural heritage domain. It deals with two standard thesauri used for storing masterpieces. The similarity measuring results of the first schema pair (test schema #1) are presented in Table 5.3.

In Table 5.3, the column represents for element names of the Schema Patient\_B, and the rows represents for elements of the Schema Patient\_A. Cells in the table represent for number of measuring elements determined for a pair of schemas. This number is  $N1 \times N2$ , where  $N1$  is the number of nodes in the first schema,  $N2$  is the number of nodes in the second schema. In order to find the most similar pair, we use the threshold value of 0.5 to categorize the similarity values. Those values are higher than threshold 0.5 are considered as similar, otherwise are non-similar. For each row of table, the algorithm will select values from 0.5 and higher, and then choose the highest similarity value.

To evaluate our proposed measuring method for similarity between elements in different XSDs (ESim), we compare ESim with the most related work (*XClust* [47]- the similarity between two DTD trees) and the similarity between XML documents (called *XMLSim*) [102]. For XML documents without schema files, we draw XML Schemas and DTD from the XML instances by using the *HIT Software* [6].

In order to answer the question of how much the information gets lost when matching two

Table 5.3: Element similarity result of the two schemas (Patient\_A and Patient\_B)

	patient	name	id	address	given	family	street	city	country	house number
patient	0.83	0.248	0.0	0.195	0.0	0.0	0.0	0.0	0.0	0.0
person name	0.165	0.527	0.0	0.275	0.275	0.33	0.33	0.33	0.33	0.33
id	0.165	0.193	0.5	0.195	0.193	0.165	0.165	0.165	0.165	0.165
address	0.33	0.193	0.28	0.875	0.193	0.28	0.28	0.28	0.28	0.28
firstname	0.028	0.195	0.195	0.193	1.0	0.83	0.275	0.275	0.275	0.275
lastname	0.028	0.195	0.275	0.028	0.83	1.0	0.275	0.275	0.275	0.275
authority	0.028	0.193	0.275	0.028	0.33	0.275	0.275	0.275	0.275	0.275
type	0.028	0.193	0.275	0.028	0.33	0.275	0.275	0.275	0.275	0.275
street	0.0	0.193	0.193	0.193	0.083	0.28	0.193	1.0	0.413	0.413
city	0.0	0.193	0.193	0.083	0.28	0.193	0.413	1.0	0.413	0.413
country	0.0	0.193	0.193	0.083	0.28	0.193	0.413	0.413	1.0	0.413
postcode	0.0	0.193	0.193	0.083	0.28	0.193	0.413	0.413	0.413	0.66

XML Schemas, we use *precision* and *recall*, which originate from information retrieval [72] and adapt to ontology matching [24]. The evaluations use the following calculations.

**Recall:** Recall specifies the share of real correspondences:

$$recall = \frac{found\_proposed\_correspondences}{all\_proposed\_correspondences} \quad (5.1)$$

**Precision:** Precision reflects the share of real correspondences among all found correspondences.

$$precision = \frac{found\_proposed\_correspondences}{all\_found\_correspondences} \quad (5.2)$$

Although precision and recall are the most widely used measures, when comparing matching systems, one may prefer to have only a single measure. Moreover, systems are not comparable based solely on *precision* and *recall*. For this reason, two measures (*F-measure* and *overall*) are introduced to aggregate the *precision* and *recall*. *F-measure* presents the harmonic mean of *precision* and *recall*.

$$F\_measure = 2 * \frac{precision * recall}{precision + recall} \quad (5.3)$$

Finally, the metric for calculating the schema matching is known as **overall**. It is defined as follows:

$$overall = recall * (2 - (\frac{1}{precision})) \quad (5.4)$$

Because of the unequal importance of Precision and Recall in Overall (see equation 5.4), we mostly use F-measure to rank the observed match quality in our discussion. Still, when appropriate, we show the values of all four measures for a comparison.

The quality measures were first determined for the single pair of schemas (to measure the similarity and then do the matching task) and then averaged over all tasks in the experiments. When

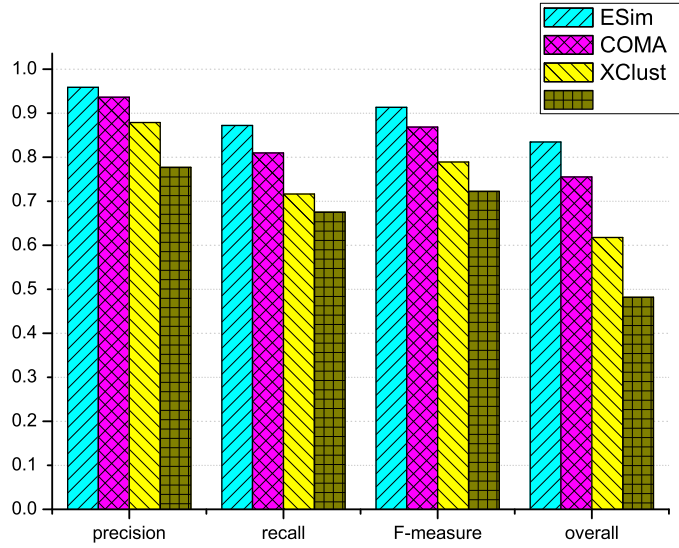


Figure 5.5: Matching comparisons of ESim to COMA, XMLSim, and XClust

talking about measuring quality through the matching task of an experiment, we refer to the average values of the measures. i.e. average Precision, average Recall, average F-measure, etc.

The comparison results among our methodology (*ESim*) and related work, such as, *COMA* [24], *XMLSim* [102], and *XClust* [47] are illustrated in Figure 5.5.

Figure 5.5 shows that our matching quality is higher than those of *XMLSim*, *XClust*, and *COMA*'s methods. The main reason is that *ESim* considers details of data type and constraint similarity whereas *COMA* focuses on linguistic name and structure; *XClust* concentrates on linguistic name, some constraints, and structure; *XMLSim* only considers the semantic name. Moreover, the element similarity measurement in *XClust* did not concern the data type similarity between two elements, whereas some element pairs have the same name, but they are different in data types.

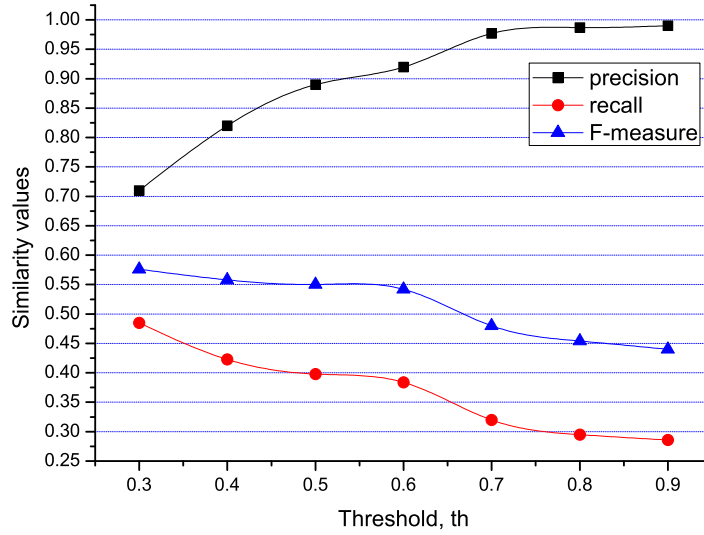


Figure 5.6: Quality of name measure

The *XMLSim* paid too much attention to the information content similarity and did not mention about the data type as well as the cardinality constraint similarity of two elements. Therefore, it gets lowest values in all calculations.

In our experiments, the threshold values are chosen between  $0.3$  and  $1$ . In order to evaluate the effect of each measuring factor to the computation results, we conduct a set of experiments calculating for each factor only and then compare their *F-measure* values with the *F-measure*'s *ESim*.

Figures 5.6, 5.7, 5.8, and 5.9 show the measuring quality of the name, data type, cardinality constraint, and structure of the elements in XSD healthcare documents. These figures indicate that there is no single measuring factor is able to determine the good correspondences.

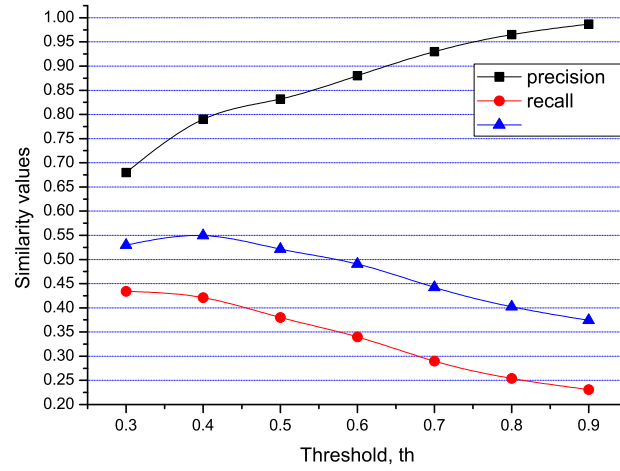


Figure 5.7: Quality of data type measure

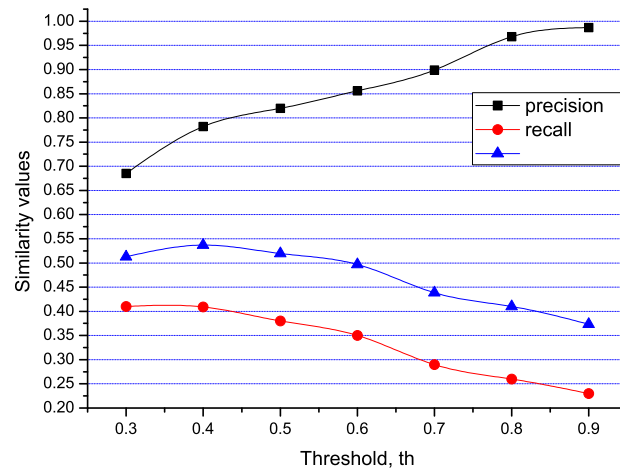


Figure 5.8: Quality of cardinality constraint measure



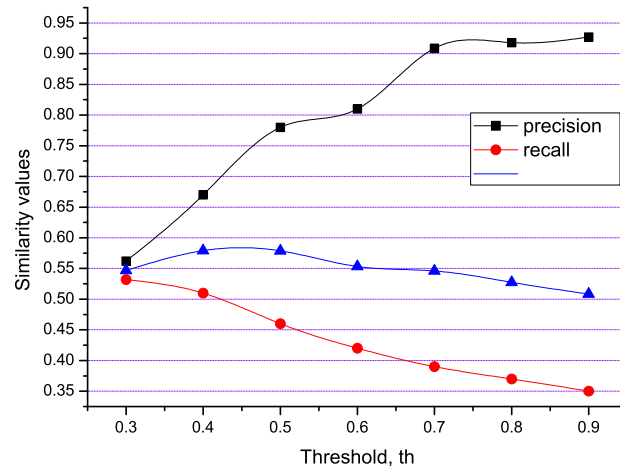


Figure 5.9: Quality of structure measure

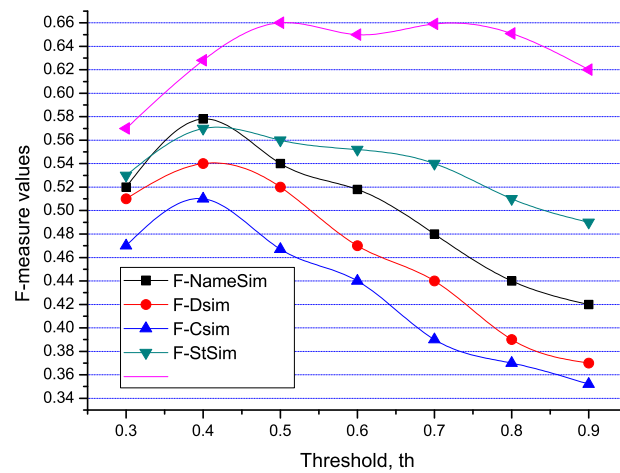


Figure 5.10: F-measure comparison

Figure 5.10 indicates that the combination of all factors gives the highest *F-measure* values, especially when we increase the threshold values. Among those factors, the structure calculation (called *StSim*) produces the greatest *F-measure* values and following by the name measurement, data type calculation and the cardinality constraint estimation.

The experimental results conducted in this study shows that although the structural resemblance plays an important role in the measure, its combination with other semantic factors produces the best matches.

## 5.2 Experiments on Duplicate Similarity in XML Transformation

### 5.2.1 Experimental setup

In this section, we describe our experimental setup. We first discuss the implementation language for transformation. Then, we present performance metrics for evaluating the quality of transformation, and then describe the real-world schemas used in the experiment.

The transformation language used in this experiment is XSLT (eXtensible Stylesheet Language Transformation) [96], [27]. XSLT is applied to an XML schema with Visual C#. We choose XSLT because its natural purpose is to act as a transformation tool for extensible languages.

We evaluate the proposed transforming strategies by matching an XSD document with an OWL ontology to determine the true matches, and compare our results with related methods. To assess the quality of the matching system, we use *precision*, *recall*, *F-measure*, and *overall*, which are presented in the previous section.

To obtain practical evidence, we applied our transformation to four healthcare XML documents: particularly, 1) *drug\_medicament.xml* [74], 2) *genertic\_risk.xml* [74], 3) *healthcaremeta-*

Table 5.4: The characteristics of the tested schemas

	Schema name	File size	# Nodes	Max depth	# Duplicates
1	drug_medicament	180 KB	683	9	0
2	patient-admission	40 KB	240	4	7
3	healthcaremetadata	5523 KB	137	7	16
4	pathology.report	328 KB	778	5	14

*data.xsd* [20], 4) *pathology.report.xml* [20]. The corresponding schemas of the test cases numbered 1), 2), and 4) are generated with a schema converter [6]. The characteristics of the four schemas are presented in Table 5.4.

Because there is no supporting tool for matching between DTD and the OWL model, in this section, we only compare our transformation of XSD into the OWL ontology with one transforming method proposed by Hannes et al. [10] and with two matching methods introduced by Toni et al. [82] and COMA++ [49].

### 5.2.2 Results

This section provides evaluation results for the test cases in Section 5.2.1. First, we conducted an experiment to determine the best threshold value for classifying duplicate elements. Second, we carried out another set of experiments to compare our work with related approaches. Lastly, we assessed the quality of each proposed measuring factor with S-Trans.

Figure 5.11 shows how error rates for classifying duplicate elements change as we use different threshold values for the DupSim measure. For the three schemas including duplicate elements in Table 5.4, we manually classified into two groups: similar and non-similar, then computed the classification error rate at each threshold (in the range 0.1 to 1.0). The weighted average

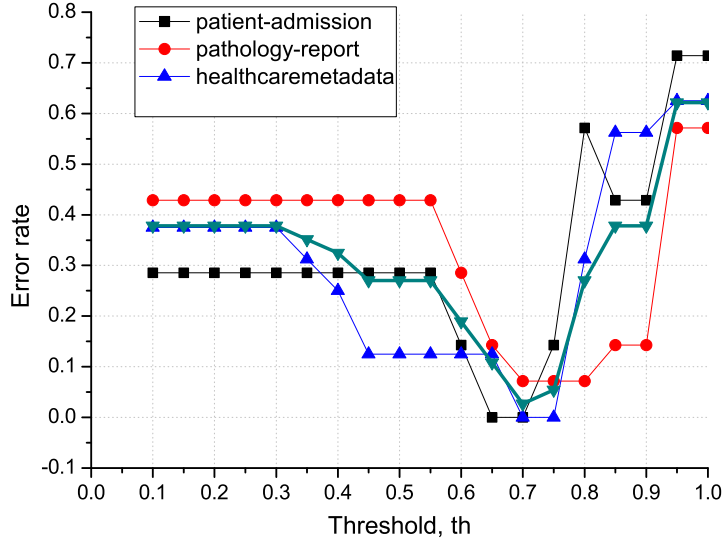


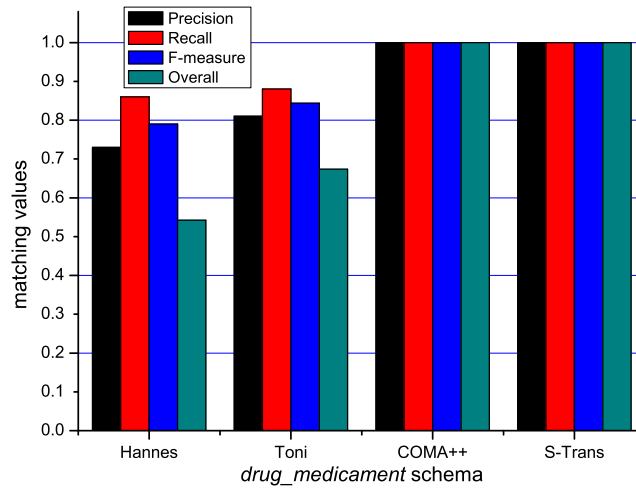
Figure 5.11: The error rate of classification at different thresholds

of the error rates for the three schemas is computed by using the number of duplicate pairs in each schema as the weighted factor. Specifically, the weighted error is determined by following Equation 5.5:

$$WeightedAverage = \frac{\sum_{i=1}^k Error(XSD_i) * n_i}{\sum_{i=1}^k n_i} \quad (5.5)$$

where  $k$  is the total number of XSD document;  $Error(XSD_i)$  is the error rate of the XSD document number  $i^{th}$ ; and  $n_i$  is the total number of duplicates in XSD document number  $i^{th}$ .

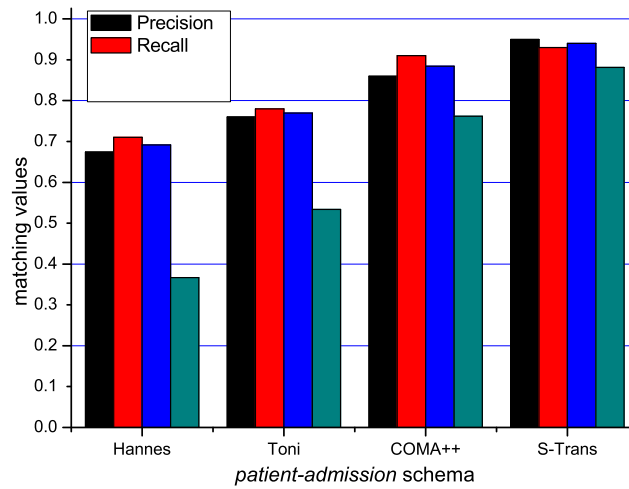
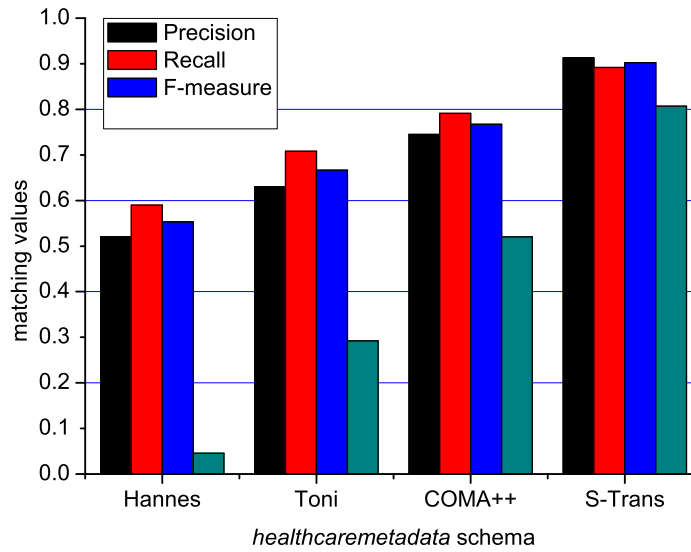
Figure 5.11 shows that very small or very large threshold values result in a large number of error rates. Particularly, at the threshold values ranging between 0.1 and 0.3, the average error rate of the classification is nearly 38%. This number decreases to approximately 27% at the threshold

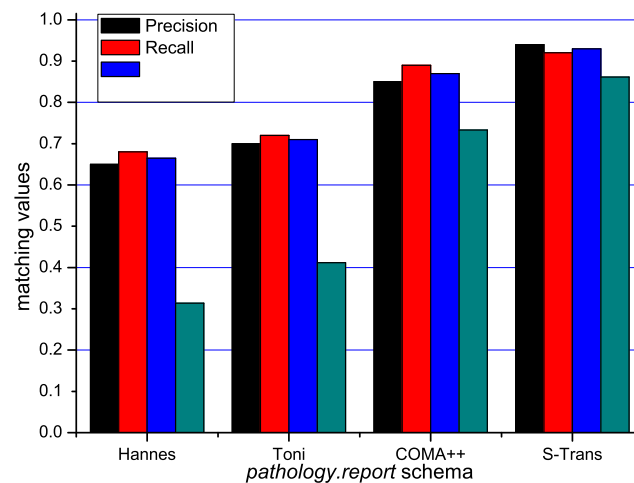
Figure 5.12: Evaluation results, *drug\_medication* schema

values between 0.45 and 0.55 and declines to 2% at the threshold value of 0.7. From the threshold values of 0.75 and higher, the error rate values steadily climb up and are highest (62%) at the threshold values of 0.95 and 1.0. Because the error rate of classification achieves the minimum value at the threshold of 0.7, we use 0.7 as the classifying value to separate the duplicates into two groups, similar and non-similar.

The comparisons of our transformation method with related work are shown in Figure 5.12, Figure 5.13, Figure 5.14, and Figure 5.15.

In Figure 5.12, S-Trans performs as well as COMA++ and outperforms Hannes and Toni's methods in terms of quality indicators. The main reason for this improved performance is that the *drug\_medication* schema does not contain any duplicate; thus, S-Trans does not change or integrate any XML element. Therefore, the COMA++ method can match 100% of XML elements

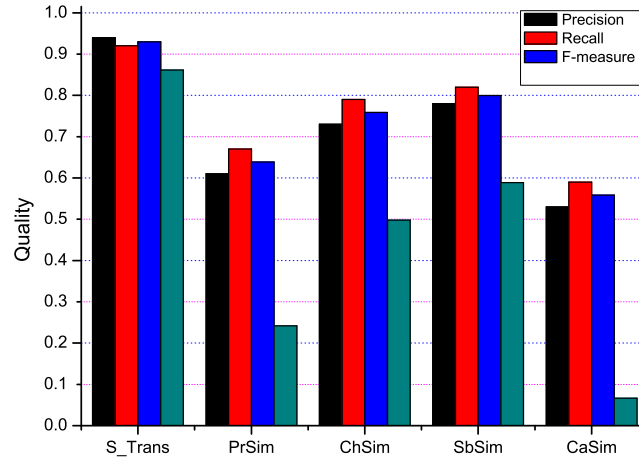
Figure 5.13: Evaluation results, *patient\_admission* schemaFigure 5.14: Evaluation results, *healthcaremetadata* schema

Figure 5.15: Evaluation results, *pathology.report* schema

with OWL concepts.

Hannes's method, however, renames all complex and simple elements by adding "has" and "dtp" prefixes, and Toni's method changes the XML elements by replacing them with the XPath expressions. Therefore, methods of Hannes and Toni result in fewer matches. However, Toni's method only changes the complex elements, whereas Hannes' method alters most of elements and properties. Therefore, Toni's approach is slightly better than Hannes' approach in terms of matching quality between XML and OWL.

In the case of larger-sized XML schema, as well as cases with more duplicate elements, S-Trans still outperforms other methods. Particularly, S-Trans's overall scores are over 84% in three test cases; see Figures 5.13, 5.14, and 5.15. Hannes' method works very poorly in test cases number 3 (Figure 5.15); its overall value is only 4% compared with 31%, 51%, and 84% of Toni,

Figure 5.16: Quality of *S-Trans*, *PrSim*, *ChSim*, and *CaSim*

COMA++, and S-Trans, respectively.

Furthermore, to determine the most important factor that affects the similarity values, we separated four similarity factors and compared them with their combination (S-Trans). The comparison results are presented in Figure 5.16. Figure 5.16 shows that *CaSim* has the lowest quality; its overall quality is only 7% compared with 24% for *PrSim*, 50% for *ChSim*, and 59% for *SbSim*. The reason for this difference in quality is that most duplicates have similar *CaSim*. However, we can also observe that the combination of all similarity factors outperforms *SbSim*; therefore, it is preferable to use multiple similarity measures instead of a single measure.



### 5.3 Experimental Summarization

In this chapter, we discussed the following thing:

- For the experiments on similarity between different schemas, ESim, we determine weighted values for *equation 3.1* and *equation 3.2*. According to the experimental result, at the values of  $\alpha$  between 0.5 and 0.65, the element similarity, ESim, has higher similarity values with user's perspective than other  $\alpha$  values, we choose the middle value, 0.55, for  $\alpha$ . Therefore, the value of the second weighted parameter is 0.45. In the same way, for *equation 3.2*, we select the value 0.4 for  $\beta$ , value 0.3 for both weighted factors:  $\chi$  and the third factor.
- We compare our proposed method, ESim, with other related work by comparing the matching results of each method. The comparison is processed on nine schema pairs, the final results are weighted average of nine schema pairs. The overall values show that, our method overcomes related approaches. Among related approaches, COMA has closest matching results with our method, the next is XClust, and XMLSim.
- We also conduct a set of experiments to determine which factor has the most effect on the similarity and the result is the children similarity factor, *ChSim*.
- For the experiments on duplicate similarity and XML transformation, S-Trans, first of all, we process an experiment to determine the classifying value by determining the error rate at different threshold. According to the experiment, at the threshold value of 0.7, the value of error rate is lowest, thus we choose 0.7 as the classifying value to categorize the duplicates into two groups: similar and non-similar. We also propose appropriate strategies to transform these two groups.
- We compare S-Trans with some related approaches, such as Hannes, Toni, and COMA++'s approaches, the results show that in case of schema without duplicate, our approach has

good quality as COMA++'s method, otherwise our approach proves the best performance with schema having duplicates.

- We also take another experiment to determine what similarity measuring factor has the most influence to the duplicate similarity value. According to the experimental result, the similarity of sibling has the most effect.

## Chapter 6

# Conclusion and Future Researches

In this chapter we bring summary of this thesis. Section 6.1.1 summaries what has been done within this thesis. Section 6.1.2 summarizes contributions of this thesis. The thesis is wrapped up with remarks about future researches in Section 6.2.

### 6.1 Conclusion

#### 6.1.1 Thesis summary

As described in the introductory chapter, the main theme of this thesis is a hybrid similarity measure for XML data integration and transformation. This theme relates two topics: Measure the similarity between different schemas for XML integration and measure the similarity of elements (particularly, duplicate elements) within a schema for XML transformation into OWL ontology. With regard to those two topics, this work covers the following aspects:

- Examination of different methods of measuring the similarity between schemas. Particularly, description of relevant similarity measure for XML integration.
- Description of similarity measure-based method for XML integration and transformation.

- Introduction of new methods for measuring the similarity between different schemas and within a schema.
- Experiments and demonstrations of new methods introduced in this thesis.

First, the thesis introduces related work to those two topics in Chapter 2. Next, Chapter 3 about similarity between different documents with the integration method and Chapter 4 about similarity of duplicates within a schema in the data transformation and Chapter 5 dealing with experimental evaluation.

Chapter 3 dealing with hybrid similarity measure based on assumption that semantics and structure are both important for similarity measure. Moreover, the XML data are usually very large and change dynamically, the measuring method should be done automatically without any user intervention. Therefore, all the similarity values should be computed by measuring metrics, not by manually.

Chapter 4 dealing with the duplicate problem in XML transformation. In related method of transforming XML data into OWL ontology, most researchers give each XML element a unique ID despite of whether this element having duplicate or not. This technique results in data redundancy, since some duplicates represent the same information, they should be transformed into an OWL concept. With the proposed measuring of duplicates similarity, we have appropriate solutions to transform those duplicates.

Chapter 5 dealing with similarity measure evaluations for both data integration and transformation. We also determine the specific values for weighted parameters in measuring metrics. Moreover, the determination of classifying value in the experiment helps the transformation of XML data become automatically and more precisely.

### 6.1.2 Contributions

Our hybrid similarity measure for XML data integration and transformation makes several contributions:

- We have proposed a hybrid similarity measure framework which considers both semantics and structure factors. This measure can be applied to compute the similarity between schemas for XML data integration or duplicate similarity within a schema for XML data transformation. This framework can operate on any type of XML data source and can execute on two types of XML schemas: XSD and DTD.
- We provide novel metrics to compute the data type similarity for attributes and the cardinality constraint resemblance between two elements. We also present a new algorithm in computing the element similarity that improves the past methods in many respects. For instance, the semantic similarity between two elements is included not only their linguistic similarity but also their data type and constraint compatibilities. The structure similarity mentions both the distance between two elements and their linguistic meaning.
- We solve the duplicate problem during transformation of XML data into OWL ontology. In the traditional approaches, the solution for duplicate problem is trivial by giving each XML element a unique identifier which results in data redundancy when duplicates represent the same information. We propose novel method to measure the similarity of duplicates in an XML schema to produce a correct, complete, and fully semantic of the transformation results.
- We propose a method to reasonably balance the role of similarity factors without user intervention. Specifically, we determine the values of weighted parameters which are previously given manually by users. Those values can be used in measuring the similarity of other XML sources.

- We propose a method to determine the classification value to categorize the duplicates into two groups: similar and non-similar. Beside the proposed transforming model for moving XML to OWL ontology, we propose a suitable strategy to transform each duplicate group into appropriate OWL concepts.
- To evaluate these measures, we conduct a set of experiments to compare our method with the related works and the important role of each measuring factor. The experimental evaluation shows that our proposed hybrid similarity measure provides the best correct similarity values of elements between schemas and duplicates within a schema in compared with related approaches.
- Moreover, we also take a set of experiments to determine which similarity measuring factor has the most influence on the final similarity result. This determination is very important in case that a system cannot measure all the factor or it wants to have the similarity quickly, it can produce the similarity values based only on the most influence factor. According to our experiments, the children similarity factor is the most influence factor to the similarity of elements between schemas, whereas the sibling similarity factor is the most effect on the duplicate similarity of duplicates within a schema.

We hope that our research are useful for both processes, XML integration and transformation, where the hybrid similarity measure is applied to produce the most correct similarity values for these tasks. If this hybrid similarity measure method is popularized, a large amount of the XML data on the Web today can become an integrated XML document or can become a meaningful OWL ontology, both results are very helpful in improving the XML data sharing between services and applications nowadays.

## 6.2 Future Researches

Directions of future research are discussed below, separately according to individual topics:

- *Measuring the similarity of different data models.* As future work we plan to extend system with a hybrid similarity measure between different documents, such as relational database and XML, XML and OWL ontology, etc.
- *Matching of different data models.* Regarding matching different data models, we plan to connect them with transformation data in terms of automatic derivation of the latter ones from the former ones. Furthermore, matching different data could also better use the similarity measure of different data models.
- *Measuring the similarity of Web pages.* We are also going to measure the similarity between Web pages by extending our proposed semantic and structural similarity metrics to measure the similarity of textual content contained within common HTML tags, the structural layout of pages, and the query terms contained within pages.

# Publications

## Journal Publications

1. **Pham Thi Thu Thuy**, Young-Koo Lee, and Sungyoung Lee, "Semantic and Structural Similarity Analysis for Integration of Ubiquitous XML Healthcare Data", Journal of Personal and Ubiquitous Computing (SCIE, IF: 1.137), ISSN: 1617-4909, 2012.
2. **Pham Thi Thu Thuy**, Young-Koo Lee, and Sungyoung Lee, S-Trans: Semantic Transformation of XML Healthcare Data into OWL Ontology, Knowledge-Based Systems Journal (SCI, IF: 1.574), ISSN: 0950-7051, 2012, <http://dx.doi.org/10.1016/j.knosys.2012.04.009>.

## Conference Publications

1. **Pham Thi Thu Thuy**, Young-Koo Lee, and Sungyoung Lee, "R2Sim: A Novel Semantic Similarity Measure for Matching between RDF Schemas", The 2012 FTRA International Conference on Advanced IT, engineering and Management (FTRA AIM 2012), Seoul, Korea, February 6-8, 2012.
2. **Pham Thi Thu Thuy**, Young-Koo Lee, and Sungyoung Lee, "Semantic and Structural Similarities between XML Schemas for Integration of Ubiquitous Healthcare Data", The 2011 FTRA International Workshop on U-Healthcare Technologies and Services (U-Healthcare 2011), Jeju, Korea, December 12-15, 2011.



3. **Pham Thi Thu Thuy**, Young-Koo Lee, and Sungyoung Lee, "XSD2RDFS and XML2RDF Transformation: a Semantic Approach", Proceedings of the 2nd International Conference on Emerging Databases (EDB 2010), pp. 167-172, Jeju, Korea, August 30-31, 2010.
4. **Pham Thi Thu Thuy**, Young-Koo Lee and Sungyoung Lee, "DTD2OWL: Automatic Transforming XML Documents into OWL Ontology", ICIS 2009, ACM, pp. 125-131, Seoul, Korea, November 24-26, 2009.
5. Brian J. d'Auriol, Tuyen Nguyen, **Thuy Pham**, Sungyoung Lee, Young-Koo Lee, "Viewer Perception of Superellipsoid-based Accelerometer Visualization Techniques", MSV'2008. pp.129 135.
6. **Pham Thi Thu Thuy**, Young-Koo Lee, Sungyoung Lee, and Byeong-Soo Jeong, "Exploiting XML Schema for Interpreting XML Documents as RDF", 2008 International Conference on Services Computing (SCC 2008), IEEE CS, pp. 555-558, 2008.
7. **Pham Thi Thu Thuy**, Young-Koo Lee, Sungyoung Lee and Byeong-Soo Jeong, "Transforming Valid XML into RDF via RDF Schema", Proceedings of the 3rd International Conference on Next Generation Web Services Practices (NWeSP'07), IEEE CS, pp.35-40, 2007.
8. **Pham Thi Thu Thuy**, Young-Koo Lee and Sungyoung Lee, "Extracting OWL Ontology from XML instances via XML Schema", The 32nd KIPS and Fall Conference, Seoul, Korea, pp. 801-802, November 2009.

# Appendix A: ESim - Evaluation Results

Because the evaluation result in the Section 5.1 is the average result of the single pair of schemas. This section presents the evaluation results of each schema pair presented in Table 5.1. The element similarity in each pair of schema is measured . All element pairs having the similarity score greater or equal the threshold 0.5 are chosen and matched together. The evaluation results of matching tasks for nine schemas, #1 to #9, which are presented in Table 5.2, in displayed in Figure A.1.

For example, for the test case #4 *Cornell vs Washington* in Figure A.1, since all the semantic and structural similarities in the given test case were correctly encoded into propositional formulas, all the quality measures of ESim reach their highest values. For pairs of test cases #1, 5, and 7, the precision of ESim performs as good as COMA and outperforms other methods.

To obtain the average result from nine pairs of test cases, we use the weighted average of nine cases. Particularly, we use the number of correct matches of each test case as the weighted factor. The precision and recall values are calculated by the following equations, A.1 and A.2:

$$precision = \frac{\sum_{i=1}^n (W_i * precision_i)}{\sum_{i=1}^n (W_i)} \quad (A.1)$$

Schema name	Evaluation factor	ESim	COMA	XClust	XMLSim
#1, patient_A, B	precision	0.900	0.900	0.643	0.563
	recall	0.750	0.643	0.450	0.429
	F-measure	0.818	0.750	0.529	0.486
	overall	0.667	0.571	0.200	0.095
#2, healthcare	precision	0.923	0.889	0.857	0.774
	recall	0.857	0.828	0.774	0.727
	F-measure	0.889	0.857	0.814	0.750
	overall	0.786	0.724	0.645	0.515
#3, Finance_Standard	precision	1.000	0.833	1.000	0.714
	recall	0.714	0.625	0.625	0.556
	F-measure	0.833	0.714	0.769	0.625
	overall	0.714	0.500	0.625	0.333
#4, Cornell_Washington	precision	1.000	1.000	1.000	0.786
	recall	1.000	0.846	0.786	0.688
	F-measure	1.000	0.917	0.880	0.733
	overall	0.786	0.724	0.645	0.515
#5, CIDX_Excel	precision	0.938	0.938	0.833	0.682
	recall	0.882	0.882	0.714	0.600
	F-measure	0.909	0.909	0.769	0.638
	overall	0.824	0.824	0.571	0.320
#6, Google_Looksmart	precision	0.966	0.950	0.950	0.826
	recall	0.934	0.905	0.877	0.792
	F-measure	0.950	0.927	0.912	0.809
	overall	0.902	0.857	0.831	0.625
#7, Google_Yahoo	precision	0.942	0.942	0.860	0.790
	recall	0.891	0.845	0.803	0.731
	F-measure	0.916	0.891	0.831	0.760
	overall	0.836	0.793	0.672	0.537
#8, Yahoo_Looksmart	precision	0.958	0.821	0.821	0.767
	recall	0.852	0.742	0.657	0.742
	F-measure	0.902	0.780	0.730	0.754
	overall	0.815	0.581	0.514	0.516
#9, Iconclass_Aria	precision	0.984	0.984	0.873	0.785
	recall	0.954	0.899	0.827	0.756
	F-measure	0.969	0.939	0.849	0.770
	overall	0.938	0.884	0.707	0.549

Figure A.1: Evaluation results of matching system for schemas in Table 5.2

$$recall = \frac{\sum_{i=1}^n (W_i * recall_i)}{\sum_{i=1}^n (W_i)} \quad (A.2)$$

where  $n$  is the number of test cases (in this experiment,  $n = 9$ );  $W_i$  is the number of correct matches of the test case number  $i$ ;  $precision_i$  and  $recall_i$  are the precision score and recall score of the test case number  $i$ .

The average results of precision, recall, and their corresponding F-measure and overall are presented in Figure 5.1.

# Appendix B: Sample of XML Schema for Transformation

Following is the full document of *pathology.report.xsd*:

Listing B.1: XML Schema for Transformation.

---

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3   <xs:element name="clinical.header"> <xs:complexType> <xs:sequence>
4     <xs:element ref="id" maxOccurs="unbounded" />
5     <xs:element ref="patient.encounter" /> <xs:element ref="provider" />
6     <xs:element ref="patient" /> <xs:element ref="codes" />
7   </xs:sequence> </xs:complexType>
8   <xs:element name="id">
9     <xs:complexType mixed="true">
10       <xs:attribute name="type" type="xs:NMTOKEN" use="optional" />
11       <xs:attribute name="authority" type="xs:NMTOKEN" use="optional" />
12     </xs:complexType> </xs:element>
13   <xs:element name="patient.encounter"> <xs:complexType> <xs:sequence>
14     <xs:element ref="id" /> <xs:element ref="practice.setting" />
```

```

15         <xs:element ref="date.time" /> <xs:element ref="location" />
16     </xs:sequence> </xs:complexType>
17 </xs:element>
18 <xs:element name="practice.setting">
19     <xs:complexType mixed="true" /> </xs:element>
20 <xs:element name="date.time">
21     <xs:complexType mixed="true" /> </xs:element>
22 <xs:element name="location">
23     <xs:complexType mixed="true" /> </xs:element>
24 <xs:element name="provider">
25     <xs:complexType> <xs:sequence>
26         <xs:element ref="person.name" /> <xs:element ref="id" />
27         <xs:element ref="addr" /> <xs:element ref="type.code" />
28         <xs:element ref="function" /> </xs:sequence>
29     </xs:complexType> </xs:element>
30 <xs:element name="person.name">
31     <xs:complexType> <xs:choice>
32         <xs:element ref="family" /> <xs:element ref="given" />
33         <xs:element ref="prefix" /> <xs:element ref="suffix" />
34     </xs:choice> </xs:complexType> </xs:element>
35 <xs:element name="addr"> <xs:complexType> <xs:sequence>
36     <xs:element ref="house.number" /> <xs:element ref="street" />
37     <xs:element ref="city" /> <xs:element ref="state" />
38     <xs:element ref="zip" /> <xs:element ref="uri" />
39     <xs:element ref="telephone" /> </xs:sequence>
40     <xs:attribute name="type" type="xs:NMTOKEN" use="required" />
41 </xs:complexType> </xs:element>
42 <xs:element name="type.code">

```

```

43     <xs:complexType mixed="true" /> </xs:element>
44 <xs:element name="function">
45     <xs:complexType mixed="true" /> </xs:element>
46 <xs:element name="patient"> <xs:complexType> <xs:sequence>
47     <xs:element ref="person.name" /> <xs:element ref="id" maxOccurs="unbounded" />
48     <xs:element ref="birth.date" /> </xs:sequence> </xs:complexType>
49 </xs:element>
50 <xs:element name="codes"> <xs:complexType> <xs:sequence>
51     <xs:element ref="coded.value" maxOccurs="unbounded" />
52     </xs:sequence> </xs:complexType> </xs:element>
53 <xs:element name="birth.date">
54     <xs:complexType mixed="true" /> </xs:element>
55 <xs:element name="city">
56     <xs:complexType mixed="true" /> </xs:element>
57 <xs:element name="clinical.body">
58     <xs:complexType> <xs:sequence>
59     <xs:element ref="clinical.history" />
60     <xs:element ref="preoperative.diagnosis" />
61     <xs:element ref="postoperative.diagnosis" />
62     <xs:element ref="operative.procedure" />
63     <xs:element ref="operative.findings" />
64     <xs:element ref="parts" />
65     </xs:sequence> </xs:complexType>
66 </xs:element> </xs:element>
67 <xs:element name="clinical.history"> <xs:complexType mixed="true" /> </xs:element>
68 <xs:element name="coded.value"> <xs:complexType mixed="true">
69     <xs:attribute name="code.system" type="xs:NMTOKEN" use="required" />
70 </xs:complexType> </xs:element>

```

```

71    <xs:element name="diagnosis"> <xs:complexType mixed="true" /> </xs:element>
72    <xs:element name="family">
73        <xs:complexType mixed="true" /> </xs:element>
74    <xs:element name="given"> <xs:complexType mixed="true">
75        <xs:attribute name="type" type="xs:NMTOKEN" use="optional" />
76        </xs:complexType> </xs:element>
77    <xs:element name="gross.description"> <xs:complexType mixed="true" /> </xs:element>
78    <xs:element name="house.number"> <xs:complexType mixed="true" /> </xs:element>
79    <xs:element name="operative.findings">
80        <xs:complexType mixed="true" /> </xs:element>
81    <xs:element name="operative.procedure">
82        <xs:complexType mixed="true" /> </xs:element>
83    <xs:element name="part"> <xs:complexType> <xs:sequence>
84        <xs:element ref="gross.description" />
85        <xs:element ref="tissue.source" /> <xs:element ref="diagnosis" />
86        </xs:sequence>
87        <xs:attribute name="name" type="xs:NMTOKEN" use="required" />
88        <xs:attribute name="frozen" type="xs:NMTOKEN" use="optional" />
89        </xs:complexType> </xs:element>
90    <xs:element name="parts"> <xs:complexType> <xs:sequence>
91        <xs:element ref="part" maxOccurs="unbounded" />
92        </xs:sequence> </xs:complexType> </xs:element>
93    <xs:element name="pathology.report"> <xs:complexType> <xs:sequence>
94        <xs:element ref="clinical.header" /> <xs:element ref="clinical.body" />
95        </xs:sequence> </xs:complexType> </xs:element>
96    <xs:element name="postoperative.diagnosis">
97        <xs:complexType mixed="true" /> </xs:element>
98    <xs:element name="prefix"> <xs:complexType mixed="true">

```



```
99      <xs:attribute name="type" type="xs:NMTOKEN" use="required" />
100    </xs:complexType> </xs:element>
101  <xs:element name="preoperative.diagnosis">
102    <xs:complexType mixed="true" /> </xs:element>
103  <xs:element name="state"> <xs:complexType mixed="true" /> </xs:element>
104  <xs:element name="street"> <xs:complexType mixed="true" /> </xs:element>
105  <xs:element name="suffix"> <xs:complexType mixed="true">
106    <xs:attribute name="type" type="xs:NMTOKEN" use="optional" />
107  </xs:complexType> </xs:element>
108  <xs:element name="telephone"> <xs:complexType mixed="true" /> </xs:element>
109  <xs:element name="tissue.source"> <xs:complexType mixed="true" /> </xs:element>
110  <xs:element name="uri"> <xs:complexType mixed="true">
111    <xs:attribute name="type" type="xs:NMTOKEN" use="required" />
112  </xs:complexType> </xs:element>
113  <xs:element name="zip"> <xs:complexType mixed="true" /> </xs:element>
114 </xs:schema>
```

---

# Appendix C: OWL Ontology Result

Following is the OWL ontology result from the *pathology.report.xsd* presented in Appendix B:

Listing C.1: OWL Ontology Result.

---

```
1 <?xml version="1.0"?>
2 <rdf:RDF
3     xmlns="http://www.owl-ontologies.com/pathology_report.owl#"
4     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
5     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
6     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
7     xmlns:owl="http://www.w3.org/2002/07/owl#"
8     xml:base="http://www.owl-ontologies.com/pathology_report.owl">
9   <owl:Class rdf:ID="pathology_report"/>
10  <owl:Class rdf:ID="patient"/>
11  <owl:Class rdf:ID="provider"/>
12  <owl:Class rdf:ID="parts"/>
13  <owl:Class rdf:ID="addr">
14    <rdfs:subClassOf> <owl:Restriction>
15      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
16        1 </owl:cardinality>
17      <owl:onProperty> <owl:DatatypeProperty rdf:ID="uri_type"/>
```

```

18     </owl:onProperty> </owl:Restriction> </rdfs:subClassOf>
19 </rdfs:subClassOf> <owl:Restriction>
20     <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
21         1</owl:cardinality>
22     <owl:onProperty> <owl:DatatypeProperty rdf:ID="type"/>
23     </owl:onProperty> </owl:Restriction> </rdfs:subClassOf>
24 </rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
25 </owl:Class>
26 <owl:Class rdf:ID="part">
27     <rdfs:subClassOf> <owl:Restriction>
28         <owl:onProperty> <owl:DatatypeProperty rdf:ID="frozen"/> </owl:onProperty>
29         <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
30             1</owl:maxCardinality>
31     </owl:Restriction> </rdfs:subClassOf>
32 </rdfs:subClassOf> <owl:Restriction>
33     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
34         0</owl:minCardinality>
35     <owl:onProperty> <owl:DatatypeProperty rdf:about="#frozen"/> </owl:onProperty>
36     </owl:Restriction> </rdfs:subClassOf>
37 </rdfs:subClassOf> <owl:Restriction>
38     <owl:onProperty> <owl:DatatypeProperty rdf:ID="name"/> </owl:onProperty>
39     <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
40         1</owl:cardinality>
41     </owl:Restriction> </rdfs:subClassOf>
42 </rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
43 </owl:Class>
44 <owl:Class rdf:ID="patient.encounter"/>
45 <owl:Class rdf:ID="clinical.header">

```

```

46 <rdfs:subClassOf> <owl:Restriction>
47   <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/
48     XMLSchema#int">1</owl:maxCardinality>
49   <owl:onProperty> <owl:DatatypeProperty rdf:ID="id_type"/> </owl:onProperty>
50 </owl:Restriction> </rdfs:subClassOf>
51 <rdfs:subClassOf> <owl:Restriction>
52   <owl:minCardinality rdf:datatype="http://www.w3.org/2001/
53     XMLSchema#int">0</owl:minCardinality>
54   <owl:onProperty> <owl:DatatypeProperty rdf:about="#id_type"/> </owl:onProperty>
55 </owl:Restriction> </rdfs:subClassOf>
56 <rdfs:subClassOf> <owl:Restriction>
57   <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/
58     XMLSchema#int">1</owl:maxCardinality>
59   <owl:onProperty> <owl:DatatypeProperty rdf:ID="authority"/> </owl:onProperty>
60 </owl:Restriction> </rdfs:subClassOf>
61 <rdfs:subClassOf> <owl:Restriction>
62   <owl:minCardinality rdf:datatype="http://www.w3.org/2001/
63     XMLSchema#int">0</owl:minCardinality>
64   <owl:onProperty> <owl:DatatypeProperty rdf:about="#authority"/> </owl:onProperty>
65 </owl:Restriction> </rdfs:subClassOf>
66 <rdfs:subClassOf> <owl:Restriction>
67   <owl:onProperty> <owl:DatatypeProperty rdf:ID="clinical.header_id"/> </owl:onProperty>
68   <owl:minCardinality rdf:datatype="http://www.w3.org/2001/
69     XMLSchema#int">1</owl:minCardinality>
70 </owl:Restriction> </rdfs:subClassOf>
71 <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
72 </owl:Class>
73 <owl:Class rdf:ID="codes"> <rdfs:subClassOf> <owl:Restriction>

```

```

74     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/
75         XMLSchema#int">1</owl:minCardinality>
76     <owl:onProperty> <owl:DatatypeProperty rdf:ID="coded.value"/> </owl:onProperty>
77 </owl:Restriction> </rdfs:subClassOf>
78 <rdfs:subClassOf> <owl:Restriction>
79     <owl:cardinality rdf:datatype="http://www.w3.org/2001/
80         XMLSchema#int">1</owl:cardinality>
81     <owl:onProperty> <owl:DatatypeProperty rdf:ID="coded.system"/> </owl:onProperty>
82 </owl:Restriction> </rdfs:subClassOf>
83 <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
84 </owl:Class>
85 <owl:Class rdf:ID="clinical.body"/>
86 <owl:Class rdf:ID="person.name">
87     <rdfs:subClassOf> <owl:Restriction>
88         <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/
89             XMLSchema#int">1</owl:maxCardinality>
90         <owl:onProperty> <owl:DatatypeProperty rdf:ID="suffix_type"/> </owl:onProperty>
91 </owl:Restriction> </rdfs:subClassOf>
92 <rdfs:subClassOf> <owl:Restriction>
93     <owl:onProperty> <owl:DatatypeProperty rdf:about="#suffix_type"/> </owl:onProperty>
94     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/
95         XMLSchema#int">0</owl:minCardinality>
96 </owl:Restriction> </rdfs:subClassOf>
97 <rdfs:subClassOf> <owl:Restriction>
98     <owl:cardinality rdf:datatype="http://www.w3.org/2001/
99         XMLSchema#int">1</owl:cardinality>
100     <owl:onProperty> <owl:DatatypeProperty rdf:ID="prefix_type"/> </owl:onProperty>
101 </owl:Restriction> </rdfs:subClassOf>

```

```

102   <rdfs:subClassOf> <owl:Restriction>
103       <owl:onProperty> <owl:DatatypeProperty rdf:ID="given_type"/> </owl:onProperty>
104       <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/
105           XMLSchema#int">1</owl:maxCardinality>
106   </owl:Restriction> </rdfs:subClassOf>
107   <rdfs:subClassOf> <owl:Restriction>
108       <owl:onProperty> <owl:DatatypeProperty rdf:about="#given_type"/> </owl:onProperty>
109       <owl:minCardinality rdf:datatype="http://www.w3.org/2001/
110           XMLSchema#int">0</owl:minCardinality>
111   </owl:Restriction> </rdfs:subClassOf>
112   <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
113 </owl:Class>
114 <owl:ObjectProperty rdf:ID="has_parts">
115     <rdfs:domain rdf:resource="#clinical.body"/> <rdfs:range rdf:resource="#parts"/>
116 </owl:ObjectProperty>
117 <owl:ObjectProperty rdf:ID="has_person.name">
118     <rdfs:range rdf:resource="#person.name"/>
119     <rdfs:domain> <owl:Class>
120         <owl:unionOf rdf:parseType="Collection"> <owl:Class rdf:about="#patient"/>
121         <owl:Class rdf:about="#provider"/> </owl:unionOf> </owl:Class>
122     </rdfs:domain> </owl:ObjectProperty>
123 <owl:ObjectProperty rdf:ID="has_addr">
124     <rdfs:domain rdf:resource="#provider"/> <rdfs:range rdf:resource="#addr"/>
125 </owl:ObjectProperty>
126 <owl:ObjectProperty rdf:ID="has_codes">
127     <rdfs:range rdf:resource="#codes"/> <rdfs:domain rdf:resource="#clinical.header"/>
128 </owl:ObjectProperty>
129 <owl:ObjectProperty rdf:ID="has_patient">

```

```

130     <rdfs:range rdf:resource="#patient"/> <rdfs:domain rdf:resource="#clinical.header"/>
131 </owl:ObjectProperty>
132 <owl:ObjectProperty rdf:ID="has_part">
133     <rdfs:domain rdf:resource="#parts"/> <rdfs:range rdf:resource="#part"/>
134 </owl:ObjectProperty>
135 <owl:DatatypeProperty rdf:about="#clinical.header_id">
136     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
137     <rdfs:domain rdf:resource="#clinical.header"/> </owl:DatatypeProperty>
138 <owl:DatatypeProperty rdf:ID="patient.encounter_id">
139     <rdfs:domain rdf:resource="#patient.encounter"/>
140     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
141 </owl:DatatypeProperty>
142 <owl:DatatypeProperty rdf:ID="city">
143     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
144     <rdfs:domain rdf:resource="#addr"/> </owl:DatatypeProperty>
145 <owl:DatatypeProperty rdf:ID="operative.findings">
146     <rdfs:domain rdf:resource="#clinical.body"/>
147     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
148 </owl:DatatypeProperty>
149 <owl:DatatypeProperty rdf:about="#coded.value"> <rdfs:domain rdf:resource="#codes"/>
150     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
151 </owl:DatatypeProperty>
152 <owl:DatatypeProperty rdf:about="#uri.type">
153     <rdfs:subPropertyOf> <owl:DatatypeProperty rdf:ID="uri"/> </rdfs:subPropertyOf>
154     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#NMTOKEN"/>
155 </owl:DatatypeProperty>
156 <owl:DatatypeProperty rdf:about="#given.type">
157     <rdfs:subPropertyOf> <owl:DatatypeProperty rdf:ID="given"/> </rdfs:subPropertyOf>

```

```

158     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#NMTOKEN"/>
159 </owl:DatatypeProperty>
160 <owl:DatatypeProperty rdf:about="#authority">
161     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#NMTOKEN"/>
162     <rdfs:subPropertyOf> <owl:DatatypeProperty rdf:ID="id"/> </rdfs:subPropertyOf>
163 </owl:DatatypeProperty>
164 <owl:DatatypeProperty rdf:ID="street">
165     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
166     <rdfs:domain rdf:resource="#addr"/> </owl:DatatypeProperty>
167 <owl:DatatypeProperty rdf:ID="prefix"> <rdfs:domain rdf:resource="#person.name"/>
168     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
169 </owl:DatatypeProperty>
170 <owl:DatatypeProperty rdf:ID="tissue.source">
171     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
172     <rdfs:domain rdf:resource="#part"/> </owl:DatatypeProperty>
173 <owl:DatatypeProperty rdf:about="#given">
174     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
175     <rdfs:domain rdf:resource="#person.name"/> </owl:DatatypeProperty>
176 <owl:DatatypeProperty rdf:ID="birth.date">
177     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
178     <rdfs:domain rdf:resource="#patient"/> </owl:DatatypeProperty>
179 <owl:DatatypeProperty rdf:ID="zip">
180     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
181     <rdfs:domain rdf:resource="#addr"/> </owl:DatatypeProperty>
182 <owl:DatatypeProperty rdf:ID="suffix"> <rdfs:domain rdf:resource="#person.name"/>
183     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
184 </owl:DatatypeProperty>
185 <owl:DatatypeProperty rdf:ID="family"> <rdfs:domain rdf:resource="#person.name"/>

```



```

186     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
187 </owl:DatatypeProperty>
188 <owl:DatatypeProperty rdf:ID="location">
189     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
190     <rdfs:domain rdf:resource="#patient.encounter"/> </owl:DatatypeProperty>
191 <owl:DatatypeProperty rdf:about="#coded.system">
192     <rdfs:subPropertyOf rdf:resource="#coded.value"/>
193     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#NMTOKEN"/>
194 </owl:DatatypeProperty>
195 <owl:DatatypeProperty rdf:ID="practical.setting">
196     <rdfs:domain rdf:resource="#patient.encounter"/>
197     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
198 </owl:DatatypeProperty>
199 <owl:DatatypeProperty rdf:about="#suffix.type">
200     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#NMTOKEN"/>
201     <rdfs:domain rdf:resource="#person.name"/>
202     <rdfs:subPropertyOf rdf:resource="#suffix"/> </owl:DatatypeProperty>
203 <owl:DatatypeProperty rdf:about="#frozen">
204     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#NMTOKEN"/>
205 </owl:DatatypeProperty>
206 <owl:DatatypeProperty rdf:ID="type.code">
207     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
208     <rdfs:domain rdf:resource="#provider"/> </owl:DatatypeProperty>
209 <owl:DatatypeProperty rdf:about="#type"> <rdfs:domain rdf:resource="#addr"/>
210     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#NMTOKEN"/>
211 </owl:DatatypeProperty>
212 <owl:DatatypeProperty rdf:about="#id">
213     <rdfs:domain> <owl:Class> <owl:unionOf rdf:parseType="Collection">

```

```

214         <owl:Class rdf:about="#patient"/> <owl:Class rdf:about="#provider"/>
215     </owl:unionOf> </owl:Class> </rdfs:domain>
216     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
217 </owl:DatatypeProperty>
218 <owl:DatatypeProperty rdf:about="#id_type">
219     <rdfs:subPropertyOf rdf:resource="#id"/>
220     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#NMTOKEN"/>
221 </owl:DatatypeProperty>
222 <owl:DatatypeProperty rdf:ID="clinical.history">
223     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
224     <rdfs:domain rdf:resource="#clinical.body"/> </owl:DatatypeProperty>
225 <owl:DatatypeProperty rdf:ID="function"> <rdfs:domain rdf:resource="#provider"/>
226     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
227 </owl:DatatypeProperty>
228 <owl:DatatypeProperty rdf:ID="preoperative.diagnosis">
229     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
230     <rdfs:domain rdf:resource="#clinical.body"/> </owl:DatatypeProperty>
231 <owl:DatatypeProperty rdf:about="#uri"> <rdfs:domain rdf:resource="#addr"/>
232     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
233 </owl:DatatypeProperty>
234 <owl:DatatypeProperty rdf:ID="gross.description">
235     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
236     <rdfs:domain rdf:resource="#part"/> </owl:DatatypeProperty>
237 <owl:DatatypeProperty rdf:ID="telephone">
238     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
239     <rdfs:domain rdf:resource="#addr"/> </owl:DatatypeProperty>
240 <owl:DatatypeProperty rdf:ID="date.time">
241     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>

```

```

242     <rdfs:domain rdf:resource="#patient.encounter"/> </owl:DatatypeProperty>
243 <owl:DatatypeProperty rdf:about="#prefix_type">
244     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#NMTOKEN"/>
245     <rdfs:subPropertyOf rdf:resource="#prefix"/> </owl:DatatypeProperty>
246 <owl:DatatypeProperty rdf:ID="clinical.header_id_type">
247     <rdfs:subPropertyOf rdf:resource="#clinical.header_id"/>
248     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#NMTOKEN"/>
249 </owl:DatatypeProperty>
250 <owl:DatatypeProperty rdf:ID="operative.procedure">
251     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
252     <rdfs:domain rdf:resource="#clinical.body"/> </owl:DatatypeProperty>
253 <owl:DatatypeProperty rdf:ID="state">
254     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
255     <rdfs:domain rdf:resource="#addr"/> </owl:DatatypeProperty>
256 <owl:DatatypeProperty rdf:ID="postoperative.diagnosis">
257     <rdfs:domain rdf:resource="#clinical.body"/>
258     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
259 </owl:DatatypeProperty>
260 <owl:DatatypeProperty rdf:ID="diagnosis">
261     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
262     <rdfs:domain rdf:resource="#part"/> </owl:DatatypeProperty>
263 <owl:DatatypeProperty rdf:ID="house.number"> <rdfs:domain rdf:resource="#addr"/>
264     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
265 </owl:DatatypeProperty>
266 <owl:DatatypeProperty rdf:about="#name"> <rdfs:domain rdf:resource="#part"/>
267     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#NMTOKEN"/>
268 </owl:DatatypeProperty>
269 </rdf:RDF>

```

---

# References

- [1] Ritesh Agrawal. Difference between OWL Lite, DL, and Full, <http://ragrawal.wordpress.com/2007/02/20/difference-between-owl-lite-dl-and-full/>, 2007.
- [2] A. Algergawy, R. Nayak, and G. Saake. Element similarity measures in XML schema matching. *Information Sciences*, 180:4975–4998, 2010.
- [3] B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Ontology-based integration of XML Web resources. In Ian Horrocks and James A. Hendler, editors, *Proceedings of 1st International Semantic Web Conference*, pages 117–131, Sardinia, Italy, June 2002. Springer.
- [4] Yuan An, Alex Borgida, and John Mylopoulos. Discovering and maintaining semantic mappings between XML Schemas and Ontologies. *Journal of Computing Science and Engineering*, 2 No. 1:44–73, 2008.
- [5] Franz Baader, Baris Sertkaya, and Anni-Yasmin Turhan. Computing the leastcommon subsumer w.r.t. a background terminology. *Journal of Applied Logic*, 5 (3):392420, 2007.
- [6] BackOffice. HIT Software - Database XML Mapping, [http://www.hitsw.com/xml\\_utilites/](http://www.hitsw.com/xml_utilites/), 2012.
- [7] Anders Berglund, Scott Boag, Don Chamberlin, Mary F. Fernandez, Michael Kay, Jonathan Robie, and Jrme Simon. XML Path Language (XPath) 2.0 (Second Edition), <https://developer.mozilla.org/en/XPath>, 2011.

- [8] E. Bertino<sup>a</sup>, G. Guerrinib, and M. Mesitia. A matching algorithm for measuring the structural similarity between an XML document and a DTD and its applications. *Information Systems Journal*, 29(1):23–46, 2004.
- [9] Philip Bille. A survey on tree edit distance and related problems. *Theor. Comput. Sci*, 337:217–239, 2005.
- [10] Hannes Bohring and Sören Auer. Mapping XML to OWL Ontologies. In Klaus P. Jantke, Klaus-Peter Fähnrich, and Wolfgang S. Wittig, editors, *Marktplatz Internet: Von e-Learning bis e-Payment*, pages 147–156, Leipzig, September 2005. GI.
- [11] Mr. Bool. E-learning xml on sql server, <http://mrbool.com/course/e-learning-xml-on-sql-server/260>, 2012.
- [12] Jon Bosak. The Role of XML in eBusiness. In *ebXML Information Day, Vienna*, pages 1–24, 2001.
- [13] Daniela F. Brauner, Alexandre Gazola, Marco A. Casanova, and Karin K. Breitman. Adaptive matching of database Web services export schemas. In *10th Int’l Conf. on Enterprise Information Systems*, 2008.
- [14] David Buttler. A short survey of document structure similarity algorithms. In Hamid R. Arabnia and Olaf Droegehorn, editors, *International Conference on Internet Computing*, pages 3–9, Nevada, USA, June 2004. Morgan Kaufmann.
- [15] Jeremy J. Carroll and Dave Turner. The Consistency of OWL Full. In *The 7th International Semantic Web Conference*, pages 1–17, 2008.
- [16] Christian Charras and Thierry Lecroq. Exact String Matching Algorithms, <http://www-igm.univ-mlv.fr/lecroq/string/>, 1997.
- [17] Sudarshan Chawathe. Comparing Hierarchical Data in External Memory. In Malcolm P. Atkinson, Maria E. Orlowska, Patrick Valduriez, Stanley B. Zdonik, and Michael L. Brodie, editors, *Proceedings of the Twenty-fifth International Conference on Very Large Data*, pages 90–101, Scotland, UK, September 1999. Morgan Kaufmann.
- [18] Claudiu Chiru, Raluca Pacuraru, and Eliza Isbasoiu. XML in E-Learning. In *The 5th International Scientific Conference eLSE “eLearning and Software for Education”*, pages 1–8, 2009.

- [19] William W. Cohen, Alex Borgida, and Haym Hirsh. Computing Least Common Subsumers in Description Logics. In William R. Swartout, editor, *Proceedings of the 10th International Conference of Artificial Intelligence*, pages 96–103, San Jose, CA, July 1992. AAAI Press / The MIT Press.
- [20] Robin Cover. The Cover Pages: Schema for Patient Medical Record, <http://xml.coverpages.org/BordenASTM20010314.html>, 2001.
- [21] Robin Cover. Health Level Seven XML Patient Record Architecture, <http://xml.coverpages.org/hl7pra.html>, 2012.
- [22] C. Cruz and C. Nicolle. Ontology enrichment and automatic population from XML data. In *4th Int'l VLDB Workshop on Ontology-based Techniques for Databases in Information Systems and Knowledge Systems*, pages 17–20, Auckland, New Zealand, August 2008. IEEE Computer Society.
- [23] T. Dalamagas, T. Chang, K.-J. Winkel, and T. Sellis. A methodology for clustering XML documents by structure. *Inform. Syst.*, 31(3):187228, 2006.
- [24] Hong-Hai Do and Erhard Rahm. COMA - a System for Flexible Combination of Schema Matching Approaches. In *VLDB*, pages 610–621, Hong Kong, China, August 2002. Morgan Kaufmann.
- [25] EGOV. XML Archive, eGovernment Resource Center , <http://www.egov.vic.gov.au/website-practice/mark-up-languages/xml/xml-archive.html>, 2012.
- [26] M. Ferdinand, C. Zirpins, and D. Trastour. Lifting XML Schema to OWL. In Nora Koch, Piero Fraternali, and Martin Wirsing, editors, *Proceedings of 4th ICWE*, pages 354–358, Munich, Germany, July 2004. Munich, Germany.
- [27] D. Foetsch and E. Pulvermeller. A concept and implementation of higher-level XML transformation languages. *Knowledge-Based Systems*, 22(3):186–194, 2009.
- [28] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using Bayesian Networks to Analyze Expression Data. In *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, page 601620, 2000.
- [29] E. Garcia. Cosine Similarity, <http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html>, 2006.
- [30] E. Garcia. Business dictionary, <http://www.businessdictionary.com/definition/>, 2012.

- [31] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199221, 1993.
- [32] T. D. Gunter and N. P. Terry. The Emergence of National Electronic Health Record Architectures in the United States and Australia: Models, costs, and questions. *Journal of Medical Internet Research*, February 2005.
- [33] Anna Harvey. XML Forms in an E-Government Architecture. In *Down to Business: Getting serious about XML*, pages 8–16, 2002.
- [34] Sandro Hawke. XML with Relational Semantics: Bridging the Gap to RDF and the Semantic Web, <http://www.w3.org/2001/05/xmlrs/>, 2001.
- [35] Longtao He, Binxing Fan, and Jie Sui. The wide window string matching algorithm. *Journal of Theoretical Computer Science*, 332:391–404, 2005.
- [36] Ian Horrocks, Bijan Parsia, Peter Patel-Schneider, and James Hendler. Semantic Web Architecture: Stack or Two Towers? In François Fages and Sylvain Soliman, editors, *Proceedings of Principles and Practice of Semantic Web Reasoning*, pages 37–41, Dagstuhl Castle, Germany, September 2005. Springer.
- [37] Ian Horrocks, Alan Ruttenberg, Sandro Hawke, and Ivan Herman. Owl working group, [http://www.w3.org/2007/owl/wiki/owl\\_working\\_group](http://www.w3.org/2007/owl/wiki/owl_working_group), 2011.
- [38] Urban Institute and Brookings Institution. The Budget Process: What is PAYGO?, <http://www.taxpolicycenter.org/briefing-book/background/budget-process/paygo.cfm>, 2008.
- [39] B. Jeong. *Machine learning-based semantic similarity measures to assist discovery and reuse of data exchange XML schemas*. PhD thesis, Department of Industrial and Management Engineering, Pohang University of Science and Technology, 2006.
- [40] Buhwan Jeong, Daewon Lee, Hyunbo Cho, and Jaewook Lee. A novel method for measuring semantic similarity for XML schema matching. *Expert Syst. Appl.*, 34(3):1651–1658, 2008.
- [41] Micheal Jervis. XML DTD vs XML Schema, <http://www.sitepoint.com/xml-dtds-xml-schema/>, November 2002.

- [42] J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference Research on Computer Linguistics X, Taiwan*, page 15 pages, 1997.
- [43] Adrovane M. Kade and Carlos A. Heuser. Matching XML documents in highly dynamic applications. In Maria da Graça Campos Pimentel, Dick C. A. Bulterman, and Luiz Fernando Gomes Soares, editors, *Proceedings of the eighth ACM symposium on Document engineering*, pages 191–198, Sao Paulo, Brazil, September 2008. ACM.
- [44] Seiji Koide and Hideaki Takeda. OWL-Full reasoning from an object oriented perspective. In Riichiro Mizoguchi, Zhongzhi Shi, and Fausto Giunchiglia, editors, *ASWC'06 Proceedings of the First Asian conference on The Semantic Web*, pages 263–277, Beijing, China, September 2006. Springer.
- [45] Gert Korthof. Information Content, Compressibility and Meaning, <http://home.wxs.nl/~gkorthof/korthof44a.htm>, 2011.
- [46] C. Leacock and M. Chodorow. Combining local context and WordNet similarity for word sense identification. In *In WordNet: An Electronic Lexical Database*, pages 265–283, 1998.
- [47] Mong Li Lee, L. H. Yang, W. Hsu, and X. Yang. XClust: Clustering XML Schemas for Effective Integration. In *ACM Press*, pages 292–299, VA, USA, November 2002. ACM.
- [48] Thomas Lee, C.T. Hon, and David Cheung. XML Schema Design and Management for e-Government Data Interoperability. *Electronic Journal of e-Government*, 7, Issue 4:381–390, 2009.
- [49] Leipzig. Database group, COMA++, <http://dbs.uni-leipzig.de/research/coma.html>.
- [50] Lus Leito, Pvel Calado, and Melanie Weis. Structure-based inference of XML similarity for fuzzy duplicate detection. In Mário J. Silva, Alberto H. F. Laender, Ricardo A. Baeza-Yates, Deborah L. McGuinness, Bjørn Olstad, Øystein Haug Olsen, and André O. Falcão, editors, *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 293–302, Portugal, November 2007. ACM.
- [51] Wei Li, Xiongfei Li, and Regen Te. Cluster dynamic XML documents based on frequently changing structures. *Advances in information Sciences and Service Sciences (AISS)*, 4(6):doi: 10.4156/AISS.vol4.issue6.8, April 2012.



- [52] Yuhua Li, Zuhair Bandar, and David McLean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Trans. Knowl. Data Eng.*, 15(4):871–882, 2003.
- [53] W. Lian, D.W. Cheung, N. Mamoulis, and Siu ming Yiu. An efficient and scalable algorithm for clustering XML documents by structure. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):82–96, 2004.
- [54] Dekang Lin. An Information-Theoretic Definition of Similarity. In Jude W. Shavlik, editor, *In Proceedings of the 15th International Conference on Machine Learning*, pages 296–304, Wisconsin, USA, July 1998. Morgan Kaufmann.
- [55] J. Madhavan, Philip A. Bernstein, An Hai Doan, and Alon Halevy. Corpus-Based Schema Matching. In Karl Aberer, Michael J. Franklin, and Shojiro Nishio, editors, *ICDE '05 Proceedings of the 21st International Conference on Data Engineering*, pages 57–68, Tokyo, Japan, April 2005. IEEE Computer Society.
- [56] J. Madhavan, S. R. Jeffery, S. Cohen, Xin (Luna), Dong, David Ko, Cong Yu, and A. Halevy. Web-scale data integration: You can only afford to pay as you go. In *Proceesings of Innovatice Data Systems Research*, pages 342–350, CA, USA, Janauary 2007. [www.crdrrdb.org](http://www.crdrrdb.org).
- [57] Marlun. XPath MDN, <https://developer.mozilla.org/en/xpath>, 2011.
- [58] Matteo Matteucci. Hierarchical clustering, <http://home.dei.polimi.it/matteucc/clustering/>, 2012.
- [59] Deborah L. McGuinness and Frank van Harmelen. Feature Synopsis for OWL Lite and OWL, <http://www.ksl.stanford.edu/people/dlm/webont/owlfeaturesynopsisjuly29.htm>, 2002.
- [60] Deborah L. McGuinness and Frank van Harmelen. Web Ontology Language (OWL Lite, OWL DL, and OWL Full) , <http://www.ksl.stanford.edu/people/dlm/webont/owlfeaturesynopsisjan22003.htm>, 2002.
- [61] Medbiquitous. Mebiquitous XML Schema, <http://ns.medbiq.org/>.
- [62] Microsoft. Learn Visual C#, Microsoft Visual Studio, <http://msdn.microsoft.com/en-gb/vstudio/hh341490>, 2012.
- [63] MSDN Microsoft. maxOccurs Property, <http://msdn.microsoft.com/en-us/library.aspx>, 2012.

- [64] Eric Miller. An Introduction to the Resource Description Framework, <http://www.dlib.org/dlib/may98/miller/05miller.html>, 1998.
- [65] MIT, ERCIM, and Keio. W3C: eXtensible Markup Language (XML), <http://www.w3.org/xml/>, January 2012.
- [66] R. Nayak and T. Tran. A progressive clustering algorithm to group the XML data by structural and semantic similarity. *Pattern Recognition & Artificial Intelligence*, 21(4):723–743, 2007.
- [67] A. Nayark and W. Iryadi. XML Schema clustering with semantic and hierarchical similarity measure. *Knowledge-Based Systems*, 20:336–349, 2007.
- [68] Nehta. XML Secured Payload Profile. Technical report, National E-Health Transition Authority, 2009.
- [69] Andrew Nierman and H. V. Jagadish. Evaluating Structural Similarity in XML Documents. In *Fifth International Workshop on the Web and Databases*, pages 61–66, 2002.
- [70] OASIS. XML.org Focus Area on eGovernment, Promoting Open Standards for Citizen-Centric Government , <http://oasis-egov.org/xmlorg-egov>, 2012.
- [71] Marek Obitko. OWL DL Semantics, <http://www.obitko.com/tutorials/ontologies-semantic-web/owl-dl-semantics.html>, 2007.
- [72] David L. Olson and Dursun Delen. *Advanced Data Mining Techniques*. Springer, 2008.
- [73] XML UK ORG. XML eGovernment, a Forum for SGML and XML Users , <http://www.xmluk.org/xml-egovernment.htm>, 2008.
- [74] OSOR. Forge Hospital, SCM: Reporitory, <http://forge.osor.eu/plugins/scmsvn/viewcvs.php/?root=hospital&sortdir=down>, 2010.
- [75] D Vint Productions. XML Schema - Data Types Quick Reference, <http://www.xml.dvint.com/>, 2008.
- [76] Peng Qin, Zhao Lu, Yu Yan, and Fang Wu. A new measure of word semantic similarity based on WordNet hierarchy and DAG theory. In *WISM '09 Proceedings of the 2009 International Conference on Web Information Systems and Mining*, page 181185, 2009.

- [77] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 19(1):17–30, 1989.
- [78] D. Rafiei, D. L. Moise, and D. Sun. Finding Syntactic Similarities between XML Documents. In *Proceedings of the 17th International Conference on Database and Expert Systems Applications*, pages 512–516, Krakow, Poland, September 2006. IEEE Computer Society.
- [79] M. Rasmussen, M. Deshpande, G. Karypis, J. Johnson, J. A. Crow, and E. F. Retzel. wCLUTO: A Web-Enabled Clustering Toolkit. Technical report, University of Minnesota, Minneapolis, MN 55455, 2003.
- [80] Philip Resnik. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, 1995.
- [81] Philip Resnik. Semantic Similarity in a Taxonomy an Information-Based Measure and its Applications to Problems of Ambiguity in Natural Language. *Journal of Artificial Intelligence Research*, 19(1):95–130, 1999.
- [82] Toni Rodrigues, P. Rosa, and J. Cardoso. Moving from syntactic to semantic organizations using JXML2OWL. *Journal of Computers in Industry*, 59:808–819, 2008.
- [83] Jan Schulz. Jaccard Similarity, <http://www.code10.info/index.php?>, 2008.
- [84] Pavel Shvaiko. ACCORD: Harmonizing, Autonomy and Coordination, [http://disi.unitn.it/ accord/](http://disi.unitn.it/accord/), 2005.
- [85] Bruno Silva and Jorge Cardoso. Semantic Data Extraction for B2B Integration. In *International Workshop on Distributed Applications for B2B Integration (DABI)*, page 16, Lisboa, Portugal, July 2006. IEEE Computer Society.
- [86] Morris Sloman. Foothill Project: Ubiquitous Healthcare, <http://www-dse.doc.ic.ac.uk/projects/ubinet/gc/manifesto/fp-healthcare.html>, 2012.
- [87] Steve Spagnola and Carl Lagoze. Edge dependent pathway scoring for calculating semantic similarity in ConceptNet. In *IWCS '11 Proceedings of the Ninth International Conference on Computational Semantics*, USA, 2011.

- [88] Australian Technical Specification. Australian Technical Specification, E-health XML secured payload profiles. Technical report, Committee IT-014, Health Informatics, 2010.
- [89] Joe Tekli, R. Chbeir, and K. Yetongnon. A hybrid approach for XML similarity. In Jan van Leeuwen, Giuseppe F. Italiano, Wiebe van der Hoek, Christoph Meinel, Harald Sack, and Frantisek Plasil, editors, *33rd conference on Current Trends in Theory and Practice of Computer Science*, pages 783–795, Harrachov, Czech Republic, January 2007. Springer.
- [90] Joe Tekli, Richard Chbeir, and Kokou Yetongnon. An overview on XML similarity: Background, current trends and future directions. *Computer Science Review*, 3(3):151–173, 2009.
- [91] P. T. T. Thuy, Y-K. Lee, and S. Y. Lee. DTD2OWL: Automatic Transforming XML Documents into OWL Ontology. In Sungwon Sohn, Ling Chen, Soonwook Hwang, Kyungeun Cho, Shigeo Kawata, Kyhyun Um, Franz I. S. Ko, Kae-Dal Kwack, Jong Hyung Lee, Gang Kou, Kiyoshi Nakamura, Alvis Cheuk M. Fong, and Patrick C. M. Ma, editors, *International Conference on Interaction Science*, pages 125–131, Seoul, Korea, November 2009. ACM.
- [92] C. Tsinaraki and S. Christodoulakis. XS2OWL: A Formal Model and a System for Enabling XML Schema Applicatopms to Interoperate with OWL-DL Domain Knowledge ans Semantic Web Tools. In Costantino Thanos, Francesca Borri, and Leonardo Candela, editors, *Proceedings of DELOS*, pages 137–146, Pisa, Italy, 2007. Springer.
- [93] International Telecommunication Union. Work related to XML and E-business, international telecommunication union, <http://www.itu.int/itu-t/e-business/mou/related/index.html>, 2002.
- [94] Princeton University. WordNet\_ a lexical database for English, <http://wordnet.princeton.edu/>, 2012.
- [95] Pim van der Eijk. Norwegian e-Health Infrastructure based on XML, ebXML and PKI. In *An OASIS Case Study*, pages 24–31, 2007.
- [96] W3C. XSL Transformations (XSLT) version 1.0, <http://www.w3.org/tr/xslt>, 1999.
- [97] W3School.com. XPath Tutorial, <http://www.w3schools.com/xpath/>, 2012.
- [98] Doug Wallace and Anthony Levinson. The XML e-Learning Revolution: Is Your Production Model Holding You Back? *Learning Solutions e-Magazine*, 5:215–236, 2006.

- [99] Kamil Wis'niewski. Anglozof.com, semantics, <http://www.tlumaczenia-angielski.info/linguistics/semantics.htm>, 2007.
- [100] Robert Worden. XML E-Business Standards: Promises and Pitfalls, <http://www.xml.com/pub/a/2000/01/ebusiness/index.html>, January, 2000.
- [101] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In James Pustejovsky, editor, *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138, New Mexico, USA, 27-30 June 1994. Morgan Kaufmann.
- [102] Dongqiang D. Yang and David M.W. Powers. Measuring semantic similarity in the taxonomy of WordNet. In James Pustejovsky, editor, *The 28th Australasian Computer Science Conference*, pages 315–322, NSW, Australia, February 2005. Australian Computer Society.
- [103] K. Zhang and D. Shasha. *Approximate Tree Pattern Matching, trees, and arrays*. Oxford University, 1997.
- [104] Ying Zhao and George Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Data Mining and Knowledge Discovery*, pages 515–524. ACM Press, 2002.