Methods and Techniques for Generation and Integration of Web Ontology Data

A Thesis Submitted for the Degree of
Doctor of Philosophy

By
Chao Wang

in

FACULTY OF INFORMATION TECHNOLOGY
UNIVERSITY OF TECHNOLOGY, SYDNEY
AUSTRALIA
JUNE 2007

UNIVERSITY OF TECHNOLOGY, SYDNEY
FACULTY OF INFORMATION TECHNOLOGY

The undersigned hereby certify that they have read this thesis entitled "**Methods and Techniques for Generation and Integration of Web Ontology Data**" by **Chao Wang** and that in their opinions it is fully adequate, in scope and in quality, as a thesis for the degree of **Doctor of Philosophy**.

Dated: June 2007

Research Supervisors: ______________________
Dr Jie Lu

____
Dr Guangquan Zhang

CERTIFICATE

Date: **June 2007**

Author: **Chao Wang**

Title: **Methods and Techniques for Generation and Integration of Web Ontology Data**

Degree: **Ph.D.**

I certify that this thesis has not already been submitted for any degree and is not being submitted as part of candidature for any other degree.

I also certify that the thesis has been written by me and that any help that I have received in preparing this thesis, and all sources used, have been acknowledged in this thesis.

[illegible]

Signature of Author

Acknowledgements

I would like to express my sincere and deep gratitude to my principal supervisor, Jie Lu, for her continuous encouragement, advice, help and invaluable suggestions. Without her support, I couldn't even start the PhD study. She is such a nice, generous, helpful and kindhearted person. I feel really happy, comfortable and unconstrained with her during my PhD study. I owe my research achievements to her professional supervision.

Many thanks are also due to my co-supervisor, Guangquan Zhang, for his valued suggestions and constant support, and for the numerous conversations with him.

I wish to thank my fellow research students and the staff of the Faculty of Information Technology, University of Technology, Sydney (UTS). Their various assistance and advice are of great benefit to this study.

I appreciate the financial support from both the UTS International Research Scholarship (IRS) and the Faculty of Information Technology Scholarship. I appreciate the travel support for attending the international conferences which I received from the Faculty of Information Technology and the UTS Vice-Chancellor's Conference Fund.

I would like to express my heartfelt appreciation to my parents for their constant support, encouragement and care over all these years. Also, I would like to thank my wife Kei Shum for her support and love during my research in the past years.

To my parents, Limin Wang and Weijuan Feng

Table of Contents

Table of Contents ix

List of Tables x

List of Figures xi

Abstract xiii

1 Introduction 1

1.1 Research Issues and Motivations 1

1.2 Contributions 5

1.2.1 Key Information Mining Method and Its Application to Web Ontology Schema Generation 5

1.2.2 Methods for Web Ontology Data Generation and Management 6

1.2.3 Web Ontology Data Matching Methods for Integration 7

1.3 Organization of the Thesis 8

1.4 Publications Related to the Thesis 11

2 Background and Literature Review 13

2.1 Introduction 13

2.2 Ontology 13

2.2.1 Definition of Ontology 14

2.2.2 Classification of Ontologies 14

2.2.3 Ontology Languages 16

2.3 Ontology Related Research Areas 19

2.3.1 Semantic Web 19

2.3.2 Web Intelligence 21

2.3.3 E-Service Intelligence 22

2.4 Ontology Generation 24

2.4.1 Ontology Schema and Data 24
2.4.2 Ontology Schema Generation 25
2.4.3 Ontology Data Generation 29
2.5 Ontology-based Data Integration and Matching 32
2.5.1 Data Semantics and Representations 33
2.5.2 Data Distribution and Management 34
2.5.3 Data Integration and Match Approaches 35
2.6 Summary 39

3 Ontology Schema Generation by Key Information Mining from the Web 41
3.1 Introduction 41
3.2 Key Information and Common Features of Web Pages 43
3.3 A Key Information Mining Method 44
3.3.1 Overview of the Method 44
3.3.2 Generation of Candidate Key Information 45
3.3.3 Entropy Evaluation 49
3.4 Experiments 52
3.5 Domain Ontology Schema Generation with KIM 53
3.6 Summary 56

4 Web Ontology Data Generation 57
4.1 Introduction 57
4.2 Web Ontology Data Conversion 58
4.2.1 Data Formats 58
4.2.2 Using XQuery for Conversion 60
4.3 Web Ontology Data Authoring 61
4.3.1 System Overview 62
4.3.2 Ontology Schema Browsing 64
4.3.3 Ontology Data Authoring 67
4.4 Discussion 70
4.5 Summary 72

5 Web Ontology Data Management 73
5.1 Introduction 73
5.2 A General Semantic Web Content Management Model 75
5.2.1 Semantic Wikis 76
5.2.2 Data focused Management 76
5.3 The Data focused SWCM Model 77

5.3.1 Semantic Contents 77
5.3.2 Participants 78
5.3.3 Operations 78
5.3.4 Rules 80
5.4 System Design and Implementation Issues 83
5.4.1 System Overview 83
5.4.2 On Implementing Operations 86
5.4.3 On Implementing the Rule Engine 87
5.5 Further Discussion 88
5.5.1 Comparison with Other Systems 88
5.5.2 Towards Semantic Web Knowledge Management 89
5.6 Summary 90

6 Ontology Data Matching by Constrained Clustering 92
6.1 Introduction 92
6.2 Scenarios 94
6.2.1 A Scenario Related to the Semantic Web 94
6.2.2 More Scenarios in Other Areas 96
6.3 Characteristics of Ontology Data 97
6.4 Duplication in Ontology Data 97
6.4.1 Notations 97
6.4.2 Constraint Rules 98
6.5 A Constrained Clustering Approach 100
6.5.1 Canopy Clustering 101
6.5.2 Canopy Clustering with Constraints 101
6.5.3 Computational Complexity 103
6.6 Experiments 105
6.7 Summary 108

7 Ontology Data Matching through Learning Adaptive Metrics 109
7.1 Introduction 109
7.2 Similarity Metrics for Instance Matching 111
7.2.1 String Edit Distance and TF-IDF 111
7.2.2 Exploring Ontology Features for New Metrics 112
7.3 Adaptive Metrics by the Learning Mechanism 114
7.4 Experiments 116
7.4.1 Data Collection 116
7.4.2 Methodology and Results 117
7.5 A Peer-to-peer Framework for Ontology Data Integration 121

7.5.1 Overview of the Framework 121
7.5.2 Ontology Data Matching in the Framework 123
7.6 Summary 124

8 Conclusions and Future Work 126
8.1 Conclusions 126
8.2 Future Work 128
8.2.1 More Functions and Applications with the KIM Method . . . 128
8.2.2 Improving Web Ontology Data Generation and Management . 129
8.2.3 Improving Ontology Data Matching Methods 129
8.2.4 Towards Applications in Real World Domains and Challenges 130

Bibliography 133

List of Tables

2.1 Semantic spectrum of ontologies 15
2.2 Different ontology learning approaches 27

3.1 Experiment results for five web sites 53

6.1 Performance of different approaches of duplicate detection 107
6.2 Precision of detection of duplicated pairs in different steps 107

7.1 Match distribution of instances in the ontology data set 117
7.2 Averaged Maximum F measure of different methods 119

List of Figures

1.1 The overall structure of the thesis 10

2.1 The ontology Spectrum according to the classification by McGuiness . 15
2.2 An example of a simple DL knowledge base consisting of a TBox and an ABox 17
2.3 Some examples of concepts, properties and individuals in OWL . . . 18
2.4 The layered architecture of semantic web proposed by Berners-Lee . . 20
2.5 Four levels of web intelligence proposed by Zhong 21

3.1 Word length in menus/navigation indicators of different web sites . . 45
3.2 Overview of the KIM method 46
3.3 Two illustrative menu subtrees 48
3.4 An algorithm for extracting a menu instance list from a web page . . 49
3.5 The interface of key information mining and OWL class marking & exportation 54
3.6 The exported raw owl file opened in Protégé for further editing and refinement 55

4.1 An XQuery used to convert XML into OWL 61
4.2 The general structure of the prototype system *robinet* 63
4.3 The interface that allows users to browse the structure of the domain ontology schema 65

4.4 The interface that shows the details of the class `Article` in the ontology schema . . . 66
4.5 The interface used for creating an instance of the class `Article` . . . 68
4.6 The created instance of the class `Article` . . . 69
4.7 The instance after related instances are created and automatically linked with it . . . 71

5.1 Fragments of an ontology schema reflecting the domain of company information management . . . 79
5.2 The general structure of the extended *robinet* system that supports SWCM . . . 84
5.3 A web page that shows the content of an instance of the class `Employee` 85
5.4 A web page showing that the value of `has-salary` can not be changed by the employee himself according to the imposed rules . . . 87
5.5 The semantic web knowledge management over content management 90

6.1 An example of the ontology data in the semantic web scenario . . . 95
6.2 An illustration of applications of different constraint rules in corresponding situations . . . 100
6.3 The algorithm of Step I . . . 102
6.4 The algorithm of Step II . . . 103
6.5 The algorithm of Step III . . . 104
6.6 The algorithm of Step IV . . . 105
6.7 The sensitiveness of precision to T_{loose} for different approaches of duplicate detection . . . 108

7.1 Recall/precision curves of different methods . . . 119
7.2 F measure scores of different methods on two major classes . . . 120
7.3 The framework that supports ontology generation and matching . . . 122

Abstract

Data integration over the web or across organizations encounters several unfavorable features: heterogeneity, decentralization, incompleteness, and uncertainty, which prevent information from being fully utilized for advanced applications such as decision support services. The basic idea of ontology related approaches for data integration is to use one or more ontology schemas to interpret data from different sources. Several issues will come up when actually implementing the idea: (1) How to develop the domain ontology schema(s) used for the integration; (2) How to generate ontology data for domain ontology schema if the data are not in the right format and to create and manage ontology data in an appropriate way; (3) How to improve the quality of integrated ontology data by reducing duplications and increasing completeness and certainty.

This thesis focuses on the above issues and develops a set of methods to tackle them.

First, a key information mining method is developed to facilitate the development of interested domain ontology schemas. It effectively extracts from the web sites useful terms and identifies taxonomy information which is essential to ontology schema construction. A prototype system is developed to use this method to help create domain ontology schemas.

Second, this study develops two complemented methods which are light weighted and more semantic web oriented to address the issue of ontology data generation. One method allows users to convert existing structured data (mostly XML data) to ontology data; another enables users to create new ontology data directly with ease.

In addition, a web-based system is developed to allow users to manage the ontology data collaboratively and with customizable security constraints.

Third, this study also proposes two methods to perform ontology data matching for the improvement of ontology data quality when an integration happens. One method uses the clustering approach. It makes use of the relational nature of the ontology data and captures different situations of matching, therefore resulting in an improvement of performance compared with the traditional canopy clustering method. The other method goes further by using a learning mechanism to make the matching more adaptive. New features are developed for training matching classifier by exploring particular characteristics of ontology data. This method also achieves better performance than those with only ordinary features. These matching methods can be used to improve data quality in a peer-to-peer framework which is proposed to integrate available ontology data from different peers.

Chapter 1

Introduction

This chapter overviews the research conducted in the thesis. It aims to help readers get a whole picture of the research topics covered in this study. It has four sections, Section 1.1 analyzes the research issues that this study tries to tackle. Section 1.2 lists the contributions of this study. Section 1.3 outlines the overall structure of the thesis. The author's publications which are related to this study are shown in Section 1.4.

1.1 Research Issues and Motivations

Data and information integration has become an ongoing research topic since computers were used to help information management. Initially, the need for data integration arose in organizations such as large enterprises where multiple databases were used to store and manage data. The research therefore was mostly related to database area, seeking both theoretical approaches [100][154] and practical solutions [89][71] so that users can be provided with a unified and consistent view of data from different data sources. Applications of data integration in this sense can obviously benefit business or industry such as financial services, telecommunications and freight brokerage [135].

While the Internet and the web have become an important platform for information publication, dissemination and acquisition, the boundary of data/information integration applications broadens. Not only commercial business and industry demand for better data/information integration and knowledge sharing. Public services like health care service require health data and information to be integrated so that the quality of services could be improved [28]. Since the event of September 11, it has been realized that appropriate data/information integration and knowledge sharing between government agencies are very essential in the battle against terrorism [26].

In the meantime, the diversity of the scenarios and the dynamic environments pose a lot of challenges for data/information integrations. Unfavorable information features (e.g., heterogeneity, decentralization, incompleteness and uncertainty) have become dramatically evident. They prevent information from being fully utilized to support various applications and services in different areas.

In response, research in the area has evolved as well. One trend is to incorporate knowledge representation techniques such as ontologies for better integration. An ontology basically is "a specification of a representational vocabulary for a shared domain of discourse - definitions of classes, relations, functions, and other objects" [65]. Obviously, this makes ontologies very suitable for the role of helping data/information integration and knowledge sharing. This role allows ontologies to have a very broad spectrum of application areas and to enrich a variety of research areas, from the general semantic web [16, 139] to some specific areas such as health informatics [133, 134]. The proposal of a semantic web treats ontologies as an important layer towards a platform where information can be understood and processed intelligently by different computers in the networks. Following that, some frameworks and systems that use ontologies as basis for integration have been proposed and developed (e.g., [7][46][81][113], more details in Section 2.5).

The basic idea of ontology related approaches for data integration is to use one or more ontology schemas to interpret data from different sources. The semantics, expressed in the ontology schema and the mapping between several ontology schemas, enable the data to be compatible and consistent with each other to reasonable degrees so that computers can process them effectively. However, several important issues come up when actually implementing the idea.

How to construct domain ontology schema(s) used for the integration.

Ontologies can actually be partitioned into two parts: ontology schemas and ontology data (more details in Section 2.2.3 and Section 2.4.1). Concepts and relations between concepts are contained in ontology schemas, which provide semantic interpretation to the data. Given a domain in which data integration is needed, constructing an ontology schema that can naturally and appropriately describe the domain is very critical to the success of the integration, especially when the process of the integration happens continually. Usually, domain experts are needed to develop ontology schemas for a particular domain and such tasks are performed almost manually, which is time-consuming. Researchers have proposed a particular paradigm which is called *ontology learning* [107], trying to semi-automatically learn from available data sources for appropriate ontology elements. Then how to specifically use this paradigm to perform domain ontology construction becomes a research question. Motivated by this paradigm, this thesis provides our particular method (Chapter 3) to this issue.

How to generate ontology data for the domain ontology schema if the data are not in the right format and to manage the ontology data in an appropriate way.

It is obvious that having an appropriate domain ontology schema is only a prerequisite step towards successful integration applications. If the domain ontology doesn't have sufficient data interpreted by its schema, then this ontology-based approach will lose its significance. Normally, data that are desirable for integration come from different sources (e.g., multiple databases, XML file, web pages, and etc). As mentioned before, integration of data from multiple databases has been studied from the perspective of database theories and practices. Typically, view-based solutions are proposed [154, 73]. In this type of solutions, views (either local schemas treated as views or global schemas as views [100]) act as a mapping so that different data sources can be queried to a more unified degree. This type of solutions can be adopted and used for integrating ontology data [24, 75]. Obviously, such data to be integrated are very structured and often static since they are stored in database. In an ever changing and growing environment such as the current web, data are not so structured. Instead, they are semi- or even un-structured data such as XML files or web pages. Using traditional methods to map such data into a unified view faces great challenges. Research on information extraction provides possible solutions to this problem. It tries to extract structured data from a large amount of raw data by learning interested patterns with typical wrapper software [97] [59] [95]. The extracted data therefore can be aligned to the domain ontology schema.

In the mean time, ontology data mapping and generation are not limited to the research area of data integration. The current web is offering a variety of ways for quick information publication. From traditional online forums and discussion groups to fashionable blogging, wikis, social bookmarking, and etc, information is created dramatically faster than ever before. However, most of such information is still oriented towards human users and therefore unstructured or semi-structured. Little semantics are expressed in a way that computers can handle just as the vision of semantic web provides. Adding semantic support (using ontologies) to these popular publication methods is desirable. And it is also very important to ensure such information is managed with a degree of reliability.

How to improve the quality of integrated ontology data by reducing duplications and increasing completeness and certainty.

In traditional data integration tasks, usually, the integrated data is aligned to a predefined global schema for easy access and analysis. But often a certain degree of overlapping happens, resulting in duplicated records that degrade the data quality and affect the further analysis. A merge/purge process [82] of duplicated records is therefore very essential to the quality of the integrated data.

Similarly, the emerging semantic web can be viewed as a process in which data published from different individuals or organizations is becoming more semantically understandable (for computers) due to the conformance to certain domain ontologies. While a particular domain ontology acting as a global schema makes data integration much easier, the duplication problem caused by overlapping data may become even more serious because of the increasing amount of sources and more decentralized environment. Therefore, the task of identification of duplicated ontology data within a certain domain becomes very necessary if we want to improve the quality of data and obtain reliable analytical results upon it or create reliable services from it (detailed discussion in Chapter 6).

In summary, the above issues for ontology related approaches for data integration motivate us to develop new methods and techniques. We are inspired to develop and use these methods to help better domain ontology schema generation in certain situations, to ease the way of generating and managing web ontology data, and to improve the ontology data integration by effective matching methods. Details of the contributions are discussed in the next section.

1.2 Contributions

The contributions of this thesis are reflected through the efforts made to tackle the issues that we've discussed before. Through the study, new methods and techniques have been developed to deal with these issues from different angles. Promising results have been achieved particularly through the experiments conducted in this study. Some of these methods have been reported in several papers that are published in international conference proceedings and journals. The following subsections will briefly introduce the major contributions presented in this thesis.

1.2.1 Key Information Mining Method and Its Application to Web Ontology Schema Generation

Developing appropriate ontology schemas for given domains is the most prerequisite step towards the integration of data in the domain. This thesis proposes a key information mining method that can be used to facilitate the development of the interested domain ontology schemas using web pages as data source. First, key information in the web pages has been identified in contrast to "noisy information". It usually refers to the certain information embedded in web pages that can categorize related web pages into several meaningful classes or identify a particular web page within a web site. By analyzing the characteristics of the existence of key information in web pages, we developed an effective method to extract potential key information from web sites and to identify those pieces of key information (such as taxonomy structures) which is essential to ontology schema construction. Our experiments with thousands of web pages from several web sites have shown quite promising results. In addition to the method itself, we also developed a prototype system that uses this method to help create domain ontology schemas.

Besides the application to ontology schema generation, the method for key information mining from web pages can also be used for some other areas. As key information is usually provided by web site constructors to keep users aware of current browsing positions in terms of topics within a web site, it is obvious that mining key information can help classify web pages precisely according to the topics. Therefore, it can contribute to the research area of web page classification. Another area that the method of key information mining can help is catalogue integration [3]. Catalogue integration is a process of merging several catalogues into a single one. It requires at least two catalogues that already have their categories. With key information extracted from web pages and these web pages from each web site categorized using our mining methods, an integration of catalogues can then proceed.

The work related to this method has been published in [162][161][167].

1.2.2 Methods for Web Ontology Data Generation and Management

In response to the issue of web ontology data generation for given domains, this study proposes two methods, trying to make the generation of web ontology data easier and more efficient. The first method is designed to deal with existing data which belong to a certain domain but have not yet been interpreted by certain ontology schemas. It goes beyond the mapping methods by converting such structured or semi-structured data into web ontology data. The use of the XQuery [19] (the query language for XML) for conversion in the method makes the process much more flexible. The other method is designed to deal with the creation of new ontology data from ordinary users (typically web users). Following this method, a web-based system "*robinet*" is developed so that users are enabled to author new ontology data collaboratively in a wiki-like style [1]. But unlike other wiki-like approaches, this system can be extended with more management functions based on the underlying ontologies. Provided that, this study also proposes an ontology enhanced model to enforce a flexible management mechanism on the system so that users can manage the web ontology data collaboratively but also securely to a certain degree.

Therefore, our *robinet* system can be used not only as a general platform to generate ontology data in a web environment, but also as a management system with a degree of security to ensure the reliability of the web ontology data.

The work related to these methods has been published in [166].

1.2.3 Web Ontology Data Matching Methods for Integration

When ontology schemas are being developed and web ontology data are being generated, the need to integrate ever growing ontology data for given domains becomes increasingly essential. Ontology data matching, as a process to improve the quality of integrated ontology data, is equally important. This study discusses several typical scenarios that demonstrate the significance of these tasks. However, on the other hand, our literature review shows that most studies on data matching are focused on the structured data stored in databases or formatted with table-like structures. While they can be used for ontology data matching in a straight forward way, we believe that the features of ontology data could allow us to develop even more effective methods based on them.

As a result, two methods have been developed in this study to perform ontology

[1]A wiki is a web site that allows visitors to add, remove, edit and change content collaboratively. Refer to http://en.wikipedia.org/wiki/Wiki for more detail

data matching for the improvement of ontology data quality when integration happens. One method uses the clustering approach. It makes use of the relational nature of the ontology data and captures different situations of matching. This results in an improvement of performance compared with the traditional canopy clustering method, which has been confirmed by our experiments. The other method goes further by using a learning mechanism to make the matching more adaptive. New features are developed for training matching classifier by exploring particular characteristics of ontology data. As a result, this method also achieves better performance than those with only ordinary features, as shown in our experiments. In addition, this study also proposes a peer-to-peer framework to integrate available ontology data from different peers. The mentioned matching methods are designed to discover matchings among the ontology data across the peers so that the overall quality of the ontology data in the framework can be improved.

The two ontology data matching methods can be further improved. They can have potential applications in the areas such as semantic web, security informatics, health care data integration, where typical scenarios have been discussed.

The work related to these methods has been published in [165][164][166].

1.3 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 provides some background knowledge of ontology and discusses related work in the area of ontology generation (including ontology schema generation and ontology data generation) and ontology-based data integration and matching. In Chapter 3, the method that is used to help develop domain ontology schema from the web is discussed. Chapter 4 then shows the methods for web ontology generation and Chapter 5 proposes the enhance model for web ontology data management. Chapter 6 starts to discuss the importance of web ontology data matching for integration with several typical scenarios and it also presents our clustering-based method for ontology data matching. Chapter 7 continues the topic of ontology matching by presenting a learning method which explores ontology features for new similarity metrics. A peer-to-peer framework that uses the ontology matching method for the improvement of the quality of web ontology data across the peers is also discussed in this chapter. Finally, the conclusions and future work are discussed in Chapter 8.

Figure 1.1 serves as a map of this thesis and shows the logical relations of all the chapters in this thesis.

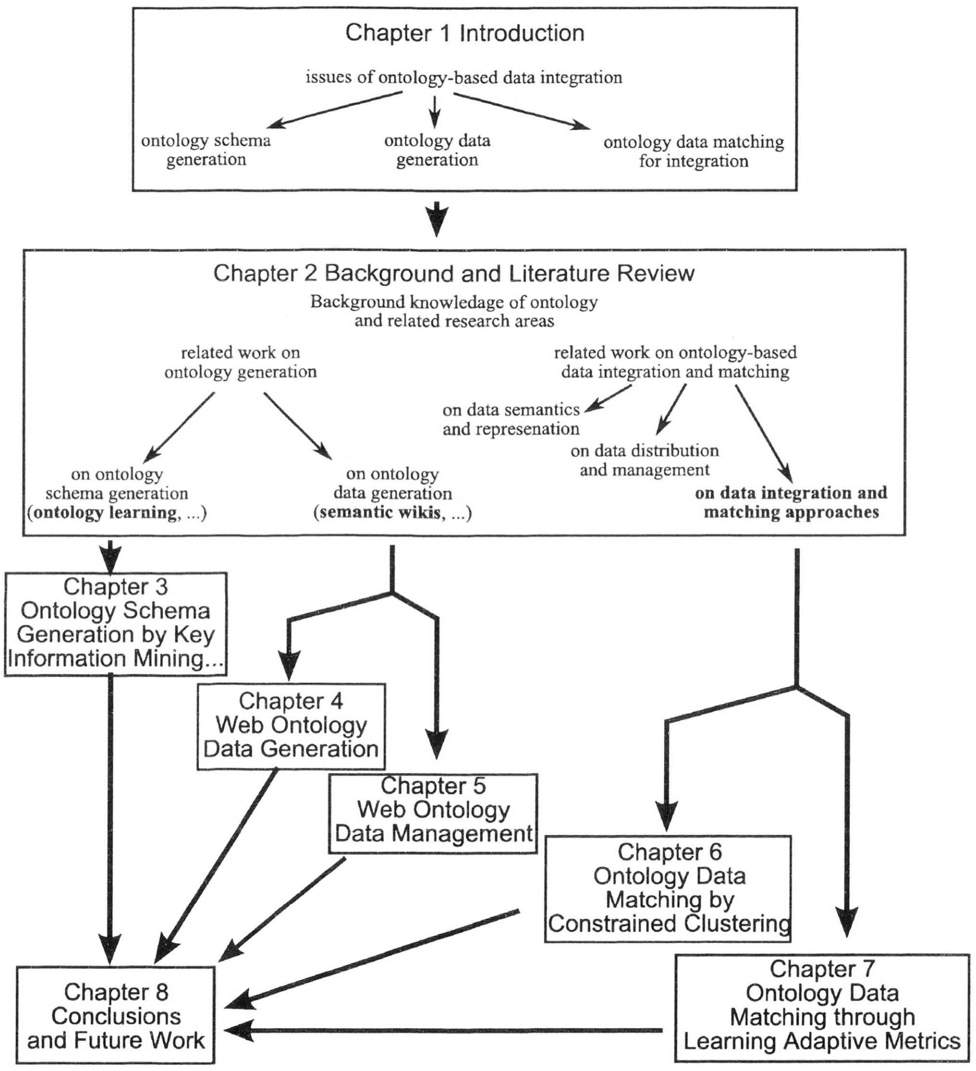

Chapter 1 Introduction
issues of ontology-based data integration
ontology schema generation
ontology data generation
ontology data matching for integration
Chapter 2 Background and Literature Review
Background knowledge of ontology and related research areas
related work on ontology generation
related work on ontology-based data integration and matching
on data semantics and represenation
on data distribution and management
on ontology schema generation (ontology learning, ...)
on ontology data generation (semantic wikis, ...)
on data integration and matching approaches
Chapter 3 Ontology Schema Generation by Key Information Mining...
Chapter 4 Web Ontology Data Generation
Chapter 5 Web Ontology Data Management
Chapter 6 Ontology Data Matching by Constrained Clustering
Chapter 7 Ontology Data Matching through Learning Adaptive Metrics
Chapter 8 Conclusions and Future Work

Figure 1.1: The overall structure of the thesis

1.4 Publications Related to the Thesis

The following is the list of related publications during the period of my PhD study in descending date order.

1. Chao Wang, Jie Lu, Guanquan Zhang and Xianyi Zeng, *Creating and Managing Ontology Data on the Web: a Semantic Wiki Approach*, *accepted by* the 8th International Conference on Web Information Systems Engineering (**WISE** 2007), 3-7 December, 2007, Nancy, France.

2. Chao Wang, Jie Lu and Guanquan Zhang, *Mining Key Information of Web Pages: a Method and its Application*, Expert Systems with Applications (**ESWA**), Volume 33, Issue 2, August 2007, Elsevier, pp.425-433.

3. Chao Wang, Jie Lu and Guanquan Zhang, *Generation and Matching of Ontology Data for the Semantic Web in a Peer-to-peer Framework*, Proceedings of the Joint Conference of the 9th Asia-Pacific Web Conference and the 8th International Conference on Web-Age Information Management (**APWeb/WAIM** 2007) (Lecture Notes in Computer Science, Springer) , 16-18 June, 2007, Huangshan (Yellow Mountains), China, pp.136-143.

4. Chao Wang, Jie Lu and Guanquan Zhang, *A Constrained Clustering Approach to Duplicate Detection among Relational Data*, Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (**PAKDD** 2007) (Lecture Notes in Artificial Intelligence 4426, Springer), 22-25 May, 2007, Nanjing, China, pp.308-319.

5. Chao Wang, Jie Lu and Guanquan Zhang, *Integration of Ontology Data through Learning Instance Matching*, Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (**WI** 2006), 18-22 December 2006, Hong Kong, China, pp.536-539.

6. Chao Wang, Jie Lu and Guanquan Zhang, *A Semantic Classification Approach for Online Product Reviews*, Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (**WI** 2005), 19-22 September 2005, Compiègne University of Technology, France, pp.276-279.
7. Chao Wang, Jie Lu and Guanquan Zhang, Mining key information of web pages, Proceedings of the 1st International Workshop on E-Service Intelligence (ESI) in conjunction with the 8th Joint Conference on Information Science (**JCIS** 2005), 21-26 July 2005, Salt Lake City, Utah, USA, pp.1573-1576.
8. Chao Wang, Jie Lu and Guanquan Zhang, *A Framework for Capturing Domain Knowledge via the Web*, Proceedings of the 11th Australasian World Wide Web Conference (AusWeb 2005), 2-6 July 2005, Gold Coast, Australia, pp.248-255.

Chapter 2

Background and Literature Review

2.1 Introduction

This chapter discusses the concepts of ontology and reviews related literature in research areas of ontology generation and ontology-based data integration and matching. First, a discussion on ontology including its definition, classification and languages is presented in Section 2.2. Section 2.3 is designated to introduce some ontology related research areas and to show the important roles that ontologies are playing in them. Section 2.4 discusses related work on ontology generation including ontology schema generation and ontology data generation. Section 2.5 reviews related work on ontology-based data integration and matching from three different perspectives. Section 2.6 summarizes this chapter.

2.2 Ontology

This section discusses the definition of ontology, the classification of different types of ontologies, and the languages that are used to express ontologies

2.2.1 Definition of Ontology

Ontology was originally a term from philosophy. The Merrian Webster online dictionary defines it as the following [1]:

> 1: a branch of metaphysics concerned with the nature and relations of being
> 2: a particular theory about the nature of being or the kinds of things that have existence

Now ontologies have been used by researchers of computer science in several research areas such as knowledge representation, artificial intelligence, natural language processing, semantic web and etc. In this sense, ontology is "a term borrowed from philosophy that refers to the science of describing the kinds of entities in the world and how they are related" [141]. Informally speaking, the most typical kind of ontology has a taxonomy and a set of inference rules [16]. In [65], Gruber states that "A specification of a representational vocabulary for a shared domain of discourse - definitions of classes, relations, functions, and other objects - is called an ontology". In [104], Maedche gives a more formal definition of an ontology structure, which is shown as follows:

Definition 2.1 (Ontology). An ontology is a 5-tuple as $O ::=< C, R, H^C, rel, A^O >$, where C is a set of classes (concepts); R is a set of relations; $H^C \subseteq C \times C$ is called taxonomy, $H^C(c_1, c_2)$ means c_1 "is-a" c_2; $rel : R \rightarrow C \times C$ is a function defined for other relations; and A^O is a logical language.

2.2.2 Classification of Ontologies

Along with the definitions, it should be noted that ontologies are being used to describe a great variety of models, ranging from simple taxonomies such as that of Open Directory Project [2] to very sophisticated models written in variants of first-order logic [21]. According to a semantic spectrum (which means a variation of levels

[1]http://www.m-w.com/cgi-bin/dictionary?va=ontology
[2]http://dmoz.org

of formalization and expressiveness) ontologies can be classified as shown in Table 2.1 and Figure 2.1 [110] [21].

Table 2.1: Semantic spectrum of ontologies

Spectrum	Description
Controlled vocabularies	Finite lists of terms
Glossaries	Lists of terms whose meaning is described in natural language, similar to dictionaries.
Thesauri	Lists of terms and definitions that standardize words for indexing purpose.
Informal is-a hierarchies	Hierarchies that use generalization relationships in an informal way.
Formal is-a hierarchies	Hierarchies that fully respect the generalization relationship
Frames	Models that include class and properties after the frame representation [115]
Ontologies with expressiveness of value restriction	Ontologies that provide constructs to restrict values their class properties can take
Ontologies with expressiveness of logic restriction	Ontologies that allow first-order logic restrictions to be expressed.

This thesis mainly focuses on ontologies with at least formal is-a features, that is, the right-hand side of the cut-off shown in Figure 2.1.

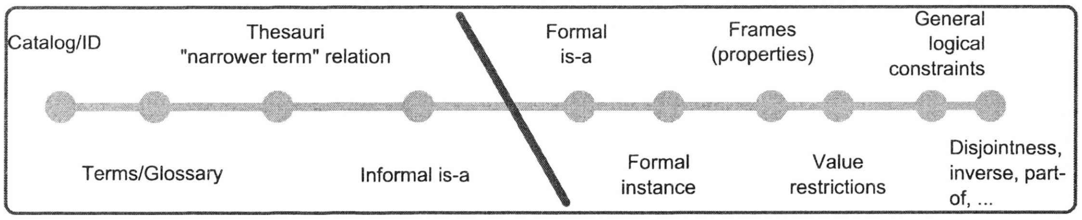


Figure 2.1: The ontology Spectrum according to the classification by McGuiness [110]

Apart from the classification based on semantic spectrum, ontologies can also be classified based on their generality, as proposed by [67]. This classification scheme divides ontologies into the following four types: Upper level ontologies; Domain ontologies; Task ontologies; Application ontologies. Obviously, upper level ontologies are the most general ontologies while application ontologies are often the most specific ontologies. This thesis mostly works on the domain ontologies and task ontologies which are of suitable generality according to the discussed topics.

2.2.3 Ontology Languages

Formal ontologies can be expressed in some forms of logic languages such as Frame logic (F-Logic) [93] or Description logics (DL) [11]. In the research area of the semantic web and many others, they are mostly expressed in DL. DL is a well-known family of knowledge representation mechanism and have been studied for several years. In DL, an ontology can be regarded as a typical knowledge base that consists of two components: a "TBox" and a "ABox". Concepts and their relations of the ontology are defined in the TBox as a set of asserted axioms. Individuals (i.e., instances of concepts) are contained in the ABox. Given a DL knowledge base, typical reasoning tasks, for example, checking subsumption of concepts, can be performed by a DL reasoner. Figure 2.2 shows an example of a simple DL knowledge base consisting of the ABox and TBox [11].

In addition to abstract logic languages, there are also some concrete languages (e.g., SHOE [81] , DAML+OIL [83] , and etc), some of which are derived from those abstract logic languages. Among them, web ontology language (OWL) [111] is a DL-based language proposed by World Wide Web Consortium (W3C). It is layered over the Resource Description Framework (RDF) [108] and RDF Schema (RDFS) [22], both of which are also proposed by W3C. Actually, OWL is a family of languages with different level of expressiveness, consisting of: OWL Lite, OWL DL and OWL Full. This design is motivated by different requirements that different users may have for a web ontology language, which can be summarized as follows (including RDF and RDFS) [147] :

- RDF and RDFS is intended for those users who primarily need a classification

$$
\begin{aligned}
\texttt{Woman} &\equiv \texttt{Person} \sqcap \texttt{Female}\\
\texttt{Man} &\equiv \texttt{Person} \sqcap \neg\texttt{Woman}\\
\texttt{Mother} &\equiv \texttt{Woman} \sqcap \exists\texttt{hasChild.Person}\\
\texttt{Father} &\equiv \texttt{Man} \sqcap \exists\texttt{hasChild.Person}\\
\texttt{Parent} &\equiv \texttt{Father} \sqcap \texttt{Mother}\\
\texttt{MotherWiouthDaughter} &\equiv \texttt{Mother} \sqcap \forall\texttt{hasChild.}\neg\texttt{Woman}
\end{aligned}
$$

TBox

$$
\begin{array}{rl}
\texttt{Father(PETER)} & \texttt{MotherWithoutDaughter(MARY)}\\
\texttt{hasChild(MARY, PETER)} & \texttt{hasChild(PETER, HARRY)}\\
\texttt{hasChild(MARY, PAUL)} &
\end{array}
$$

ABox

Figure 2.2: An example of a simple DL knowledge base consisting of a TBox and an ABox [11]

hierarchy;

- OWL Lite adds the possibility to express (in)equalities and simple constraints on such hierarchies;
- OWL DL supports those who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time);
- OWL Full is meant for those who want maximum expressiveness and syntactic freedom with no computational guarantee.

OWL provides a set of vocabulary as constructs, enabling people to define concepts, properties, individuals, and their relations. Typically, property in OWL has two categories: data type property and object property. Data type properties allow people to describe specific attributes of a concept, such as "age of person", "address of company". Meanwhile, object properties enable people to link two concepts with a semantic relation, like "supervision" between "professor" and "student". Related examples are shown in Figure 2.3.

```
...
<owl:Class rdf:about="#FullProfessor">
  <rdfs:suClassOf rdf:resource="#Professor"/>
</owl:Class>

<owl:ObjectProperty rdf:about="#hasSupervisor">
  <rdfs:domain rdf:resource="#Student"/>
  <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:about="#age">
  <rdfs:domain rdf:resource="#Person"/>
</owl:DatatypeProperty>

<u:GraduateStudent rdf:about="#GraduateStudentA">
  <u:name>Graduate Student A</u:name>
  <u:age>30</u:age>
  <u:memberOf rdf:resource="#UniversityB" />
  <u:hasSupervisor rdf:resource="#ProfessorC" />
  ...
</u:GraduateStudent>
...
```

Figure 2.3: Some examples of concepts, properties and individuals in OWL

This thesis will discuss ontologies expressed in DL languages and the DL-based OWL language in most cases. As DL languages with high level of expressiveness will cause dramatic computational costs, they are not very often used in solving practical problems. Therefore, ontologies used in this thesis will be kept to an acceptable level of expressiveness.

2.3 Ontology Related Research Areas

As ontologies are mostly used to provide ways to describe entities and their relations, the fundamental role of ontology related techniques is to help data/information integration and knowledge sharing. This role allows ontologies to have a very broad spectrum of application areas and to enrich a variety of research areas, from the general semantic web [16, 139] to some specific areas such as electronic commerce [57, 62], health informatics [133, 134] and etc. According to the topics of this thesis, we briefly introduce the following several research areas and the specific roles of ontologies in these areas.

2.3.1 Semantic Web

The semantic web is an extension of the current world wide web, in which information is given well-defined meaning, better enabling computers and people to work in cooperation [16]. The major concern in realization of the semantic web is the creation of appropriate languages which are suitable for the delivery of semantic web contents. This may be inspired by the great success of hypertext markup language (HTML) [132] in creation of the massive web. As HTML is mostly an presentation language rendered by web browsers so that human can read and understand the contents it delivers, it can hardly fulfill the goal of a machine-understandable web.

The picture of a layered stack shown in Figure 2.4 outlines how the semantic web could be formed [15]. Unicode [3] and Universal Resource Identifier (URI) construct the foundation of this stack. Unicode provides a unique number for every character, no matter what the platform, the program, or the language is. This facilitate the processing of characters, especially when characters from different languages need to be encoded. The Unicode Standard has already been adopted by many industry leaders [86]. In the meantime, a URI is a compact sequence of characters that identifies an abstract or physical resource [14]. Since URIs have global scope and are interpreted consistently across contexts, associating a URI with a resource means that anyone

[3]http://unicode.org/

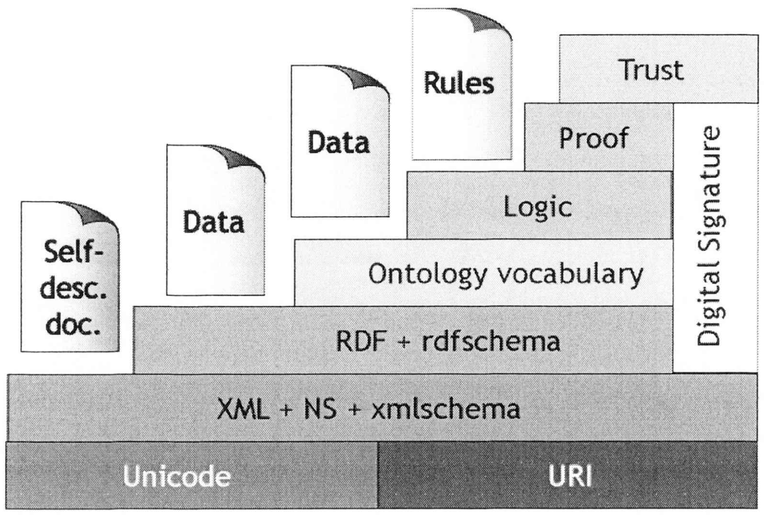


Figure 2.4: The layered architecture of semantic web proposed by Berners-Lee [15]

can link to it, refer to it, or retrieve a representation of it. Therefore, they underpin the Semantic Web, allowing machines to process data directly [139].

Built upon the layer of Unicode and URI, is Extensible Markup Language (XML), XML Name Space (NS) and XML Schema. XML [20] is a simple, very flexible text format, designed originally to meet the challenges of large-scale electronic publishing. It is also playing an increasingly important role in the exchange of a wide variety of data on the web and now on the semantic web.

Resource description framework (RDF) [108] and RDF Schema (RDFS) [22] were among the first attempts to bring semantics to the web. They are usually encoded in XML syntax as they are on top of the layer of XML. They have provided a minimal ontology representation language that the research community has adopted fairly widely [139].

RDF and RDF Schema may not be sufficient enough for people to specify ontology vocabulary. For those who need greater expressivity for ontology development and deployment, the Web Ontology Language (OWL) specification has been proposed by by World Wide Web Consortium (W3C). As we have mentioned before, this ontology language has become more and more popular compared to other ontology languages.

When ontologies become widely used for rendering rich semantics to data, it is possible to develop rules and inference mechanisms to take advantage. Highly intelligent web applications and services could be developed, which finally turns the vision of the semantic web into reality.

From the discussion, the important position of ontologies in the formation of the semantic web is obvious.

2.3.2 Web Intelligence

Compared to the research of the semantic web, which involves many efforts in realizing a machine-understandable web, web intelligence (WI) is a research area that exploits artificial intelligence (AI) and advanced information technology on the web and internet [174]. In [176], Zhong proposes four conceptual levels of web intelligence in order to develop a "wisdom web". The four-level framework ranges from the lower hardware-centered level to the higher application-centered level as shown in Figure 2.5. It builds upon the fast development of application of various hybridization of AI and web technologies.

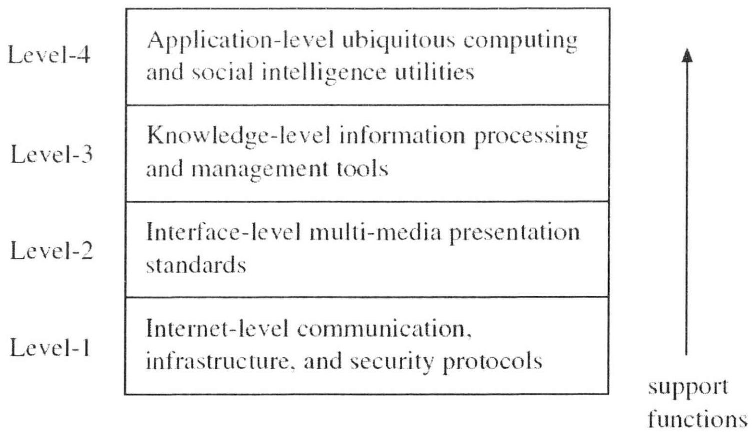

Level-4
Application-level ubiquitous computing and social intelligence utilities
Level-3
Knowledge-level information processing and management tools
Level-2
Interface-level multi-media presentation standards
Level-1
Internet-level communication, infrastructure, and security protocols
support functions

Figure 2.5: Four levels of web intelligence proposed by Zhong [176]

According to Zhong [176], one of the fundamental issues of WI is to study the semantics on the web with respect to level 3 of WI, that is, modeling semantics of web information. This may look similar to what the semantic web tries to address.

These two research areas (WI and semantic web) may overlap each other to some extent. It seems that web intelligence has an emphasis on the exploration of artificial intelligence in developing advanced web-based applications. However, to develop web-based applications with high-level intelligence, the dependence on the shared knowledge among machines and users is inevitable. This again relies on the research and development of ontologies which bring semantics to the web. Therefore, the roles and functions of ontologies in this situation includes: communication between different web communities; agent-based communications; knowledge-based web retrieval; understanding web contents and intelligent discovery of social networks and web communities [176].

2.3.3 E-Service Intelligence

Many business organizations and government departments are developing and providing Internet based electronic services (e-services) featuring various intelligent functions. This form of e-services is commonly called *e-service intelligence* (ESI) which, integrates intelligent technologies and methodologies into e-service systems for realizing intelligent Internet information searching, presentation, provision, recommendation, online system design, implementation, and assessment to Internet users [103]. Similar to the research of WI, one important element of ESI is the integration of intelligent technologies. But it put more emphasis on the improvement of particular electronic services provided by various organizations.

The use of ontologies can help improve the performance of e-Services. For example, Using informal ontologies such as those of Open Directory Project [4] or eBay Catalogs [5], can help users locate and navigate information on the web sites more efficiently. When it comes to providing recommendation and personalization services to web users, ontology-enhanced semantic web mining could achieve more sensible results than traditional techniques [39] [40]. Incorporating ontology technologies into

[4]http://dmoz.org
[5]http://www.ebay.com

web services [38] [5] creates the new research field of *semantic web services*, which "enables a wide variety of agent technologies for automated web service discovery, execution, composition, and interoperation" [112].

Here we provide a typical case to show the potential benefit of using ontology in e-Services. Nowadays, while more and more people are opting to buy products online, online product reviews have therefore begun to be an importance source for people's online purchase decision making. However, the increasing amount of reviews makes it impossible for users to read though all of them. Various techniques [85][43][128][163] have been employed to automatically analyze online reviews so that integrated information like classification of positive reviews and negative reviews can be generated for customers' references. For example, Pang, et al [128] investigate several supervised machine learning methods (naive bayes classifier, maximum entropy classifier and support vector machines) to semantically classify movie reviews with the conclusion that machine learning techniques outperform human-produced baseline but do not perform as well on sentiment classification as on traditional topic-based classification. Dave, et al [43] develop a method for automatically classifying positive and negative reviews and experiment several methods related to feature selection and scoring. Hu and Liu [85] use the adjective synonym sets and antonym sets in WordNet [55] to judge semantic orientations of adjectives, which helps to classify the semantic orientations of product reviews with improved accuracy. Wang, et al [163] develop a method based on machine learning to judge the semantic orientations of reviews at the sentence level.

However, the unstructured nature of online product reviews often degrades the performance of those methods, which results in an unsatisfactory service of product review summarization. If an appropriate domain ontology is available and is able to depict the features of products that customers intend to comment on and the possible vocabulary with their semantic categories that customers intend to use to express opinions, the task of online product reviews integration would be easier and better electronic services such as product recommendation, personalized shopping, and etc could be delivered.

2.4 Ontology Generation

The previous section clearly shows the important roles of ontologies in various research areas. Ontology generation (or ontology construction) therefore becomes very crucial.

Generation of ontology can start from different points, such as: 1) starting from scratch, 2) starting from existing ontology, and 3) from a corpus of information sources [156] . Along with these different starting points, the degrees of automation in generating ontology vary, ranging from fully manual, semi-automated, to fully automated. Fully automated ontology construction may only used for very lightweight ontologies in limited circumstances [156].

From the view of engineering, methods to generate an ontology could be summarized the following general types: (1) bottom up - from specification to generalization; (2) top-down - from generalization to specification; and (3) middle-out - from the most important concepts to generalization and specification [50].

Unlike many other literatures, this thesis distinguishes the parts of ontologies (schema and data) and discusses the generation of them respectively.

2.4.1 Ontology Schema and Data

As mentioned earlier in Section 2.2.3, ontologies expressed in DL have two parts: TBox and ABox [11]. Similar to the notions in database research field, *ontology schema* can be used to refer to the concepts, relations and etc in the TBox, while *ontology data* can be used to refer to the individuals or instances of concepts in the ABox. These notions seems to be common especially when OWL is used to encode ontologies.

In some literatures when ontology is mentioned, it sometimes or mostly refers to ontology schema. Accordingly, ontology data is often mentioned as "ontology annotated/aligned data". In the DL research community, ontology data (i.e. instances of concepts) is often mentioned as "individuals". In this thesis, unless specific notes are provided, "instance" is used as an equivalent term to "individual".

2.4.2 Ontology Schema Generation

This subsection discusses the traditional ways of creating ontology schemas manually with tools, analyzes the new paradigm of ontology learning and its various methods, and finally focuses on the study of ontology learning from web pages.

Manual Ontology Schema Generation with Tools

Firstly, ontology schemas can be generated manually by some experts who are familiar with those interested domains and with ontology technologies as well. There exists several software programs that can be used for design and development of ontology schemas as well as ontology data. To list a few, they are Protégé [60], OntoEdit [148], SWOOP [92]. Protégé [6] is a free, open source ontology editor and knowledge-base framework that supports two main ways (Protégé-Frames and Protégé-OWL) of modeling ontologies including ontology schemas. OntoEdit [7] provides an ontology engineering environment that supports the development and maintenance of ontologies (typically ontology schemas) using graphical means. In contrast to the traditional knowledge representation (KR)-based paradigms which Protégé and OntoEdit belong to, SWOOP [8] is a "hypermedia-based OWL ontology browser and editor that is meant for rapid and easy browsing and development of OWL ontologies". The graphic user interface and proper reasoning facilities in those software tools help experts to create concepts, properties and etc intuitively and consistently. Associated with the manual construction, often are some ontology design principles or methodologies such as those described in [155].

[6]http://protege.stanford.edu/
[7]http://www.ontoknowledge.org/tools/ontoedit.shtml
[8]http://www.mindswap.org/2004/SWOOP/

Ontology Learning: a Class of Different Methods

Obviously, the manual ontology schema generation is often time-consuming, labor-intensive and error-prone, especially when working on large scale ontologies. These drawbacks force researchers to consider adding certain degrees of automation in this manual process. This results in a typical research topic called *ontology learning* [9], which involves utilizing number of complementary disciplines and methods to process different types of data sources (ranging form unstructured, semi-structured, to fully structured) for the support of semi-automatic, cooperative ontology engineering [107].

In the work of Maedche et al [107], a general ontology learning framework is proposed including a processing cycle from ontology import and reuse, extraction, pruning, refinement, to evaluation and an architecture with components like management component and algorithm library. Besides the general framework and architecture, some implemented techniques are also discussed to help ontology learning from free text, dictionaries and legacy ontologies.

In the work of Gómez-Pérez et al [63], a fairly comprehensive survey of different methods and techniques is delivered. It summarizes different ontology learning approaches and related tools with respect to the different types of input: text, dictionary, knowledge base, semi-structured schemata and relational schemata. Table 2.2 shows a survey of different ontology learning approaches based on the survey of [107] and [63].

According to Table 2.2, many different methods have been applied to perform ontology learning from different sources. However, the purposes of using these methods differ. Many of these research works (e.g., [80][1][53][4] listed in the table) already have a basic domain ontology. Their proposed methods try to enrich this basic ontology by adding more related vocabulary as concepts and relations to the existing ontology, which does not change major concepts and structures but makes the ontology more complete and precise [127]. Some research works (e.g., [131][30][31][169]

[9]Although it's called ontology learning, the major tasks in this topic are to learn or enrich concepts and relations from available data sources. So it is regarded as a class of methods for ontology schema generation

Table 2.2: Different ontology learning approaches.
(Abbreviation explanations: EO: Enrich existing ontologies; DT: Develop taxonomies for ontologies; CR: Discover conceptual relations; DI: Data integration; TE: Thesaurus enrichment)

Sources	Approaches	Main goals	References
Free text and semi-structured text	Statistical and Frequency based approaches	EO, DT	[1] (EO), [53] (EO), [42] (DT), [138] (EO)
	Clustering	EO, DT	[1] (EO), [54] (DT), [30] (DT)
	Formal Concept Analysis	DT	[131], [31], [169]
	NLP and word patterns	EO	[72], [80], [4], [119], [138]
	Association Rules	CR	[106]
Semi-structured schema	machine learning	DI	[51]
relational schema	mapping	DI	[136], [146]
Dictionary	linguistic patterns, statistical approaches	TE	[79], [88]
Knowledge base	Statistical approaches	DT	[149]

listed in the table) , on the other hand, don't have a basic ontology to start with, but need to construct it from scratch with available sources. The latter seems to be more difficult than the former.

Ontology learning from free texts or semi-structured data is more popular than from other sources according to Table 2.2. This may be largely because of the widely available sources of free texts and semi-structured texts, especially over the web. Since there is lack of formal syntax and restricted structure in free texts, learning ontologies from them is quite difficult. Some basic statistical methods (e.g., [1][53] listed in the table) can only help suggest popular terms from the source texts as possible ontology concepts or relations. More advanced methods use techniques such as natural language processing, formal concept analysis, machine learning and etc to perform learning tasks, trying to generate more than suggested concept and relationship terms. Semi-structured texts largely refer to HTML pages from the web. The tags and structures in them sometimes have typical patterns, which can help improve the performance of ontology learning.

Learning Domain Ontologies from Web Pages

Our research focuses on learning domain ontologies from web pages. So in the following, we will discuss the most related work in more detail.

Maedche et al. [105] focus on the development of the taxonomic backbone of the ontology (an essential part of ontology engineering) by investigating existing work on learning taxonomic relations from texts that are extracted from a web site. Two types of existing approaches (symbolic and statistics-based approaches) are analyzed including their advantages and disadvantages. For a reconciliation between the two types, it proposes new algorithms for taxonomy learning using existing taxonomic relations as background knowledge. Experiments are conducted to evaluate the performances of the approaches including the proposed approaches. Although texts from the web are used as the sources for ontology learning in this study, the features of such hypertexts (HTML tags and structures, links, etc) are not explored in the proposed approaches. Like many other studies, texts from the web are often preprocessed by removing all the HTML tags. The elimination of HTML tagging information makes such texts just look like normal free texts, which prevents the possibility of developing more sophisticated methods that can make use of HTML tagging information.

OntoMiner [42][45] is a more specific system able to bootstrap and populate ontologies (including the backbone taxonomy for the ontology schemas) from domain-specific web sites. In this system, HTML tagging information is used intensively to capture the desired information (concepts and hierarchical relations) in the web pages. Both its semantic partitioning algorithm and tree-mining algorithms rely on the occurring patterns of HTML tags. Therefore, the results are quite promising thanks to the sensible use of such tagging information in the web pages. However, the way it uses to capture the vocabulary and structures in web pages is quite complicated and it may need to tune many parameters to obtain such the desired information precisely.

In the meantime, the patterns of HTML tagging information have been widely used in various web mining tasks, including automatic information extraction [97] [59] [95], table and block detection [168] [102] [27] [142], classification and clustering [101] [175], and etc. Such tagging information helps improve the performances of these tasks. This leads us to believe that ontology learning from semi-structured web pages should benefit from the exploration of the tagging information. This has actually been confirmed by our study (presented in Chapter 3).

2.4.3 Ontology Data Generation

Similar to ontology schema generation, there are various ways to generate ontology data. This section lists five main techniques associated with this topic.

Ontology Development Tools

Some ontology development tools (e.g. Protégé [60], OntoEdit [148], SWOOP [92]) can be used to assist create new ontology data as well as to develop ontology schemas. However, as these tools are designed with an emphasis on knowledge representation (i.e. design of a proper ontology schema that reflects a certain domain), they are often used by experienced domain experts familiar with ontology related techniques. Ordinary users may have difficulty using it, which hinders the generation of ontology data in a large scale.

Mapping Using Views

From the perspective of traditional data integration research, the task of ontology data generation can be performed by creating mappings between existing data schema (mostly those schema describing structured data stored in databases) and the target domain ontology schema [24, 75]. The mappings, sometimes regarded as views, allows existing structured data to be treated as aligned ontology data for certain domains. If such structured data are available, this type of solutions is quite efficient for generating ontology data. However, it is difficult to use this type of solution to deal with unstructured or semi-structured data, such as web pages.

Annotation

Annotation of existing data (mostly web pages) with ontologies is also a way to generate ontology data. SHOE [81], Ontobroker [46], CREAM [78] are the systems that can be used to create meta information as ontology data for existing web pages by users. Normally, a crawler is needed to gather these ontology data from annotated web pages scattered over the web.

Ontology data can also be generated by automatically annotating web pages (e.g. [48, 32, 29]). Although a large amount of ontology data can be generated by this type of methods, some applications may not be able to use them due to its relatively poor quality.

Semantic Wikis

Recently, inspired by the success of wiki-like systems [10], researchers have shown increasing interest in bringing wiki systems and semantic web technologies together [96]. This results in the proposal of semantic wikis, which tries to enable people to create ontology data through online collaboration.

The main idea of most wiki systems is to let web users create and maintain web contents in a collaborative way. These systems provide convenient functions so that an ordinary user with little knowledge of web page making can contribute contents to the web. Wikipedia[11], the most well-known online collaborative encyclopedia on the web, is a successful application of wiki systems. However, most contents maintained in wiki systems are semi-structured web pages only for web users to search and read.

[10]http://en.wikipedia.org/wiki/Wiki
[11]http://www.wikipedia.org

Facts and knowledge expressed in those web pages lack proper syntax and semantics for computers to process.

Semantic wiki systems then emerge to introduce semantic web standards (e.g., RDF/RDFS , OWL) and techniques to the common wiki systems, trying to make the collaborated contents have more semantics for computers to understand and process. We will discuss these systems as follows.

Platypus Wiki [151] is an extended wiki system that allows users to input RDF and OWL statements in addition to plain text. However, how the system helps users to create such semantic statements has not been discussed in detail in [151].

Rhizome [143] is an application stack that lets developers build web applications which uses familiar technology such as XML and etc, but run in an environment where the outside world is presented as a universe of RDF statements. It can be used to develop a wiki-like application that lets the user create arbitrary RDF resources and the application logic for presenting and interacting with them. Several custom text formats (ZML, RxML) are used in the system to write semantic content. The use of those custom text formats may help experienced users to generate semantic content effectively and allow the system to process it efficiently, but it can also raise some barriers for ordinary users.

Powl [9] is a web based platform that provides a collaborative environment for semantic web development. OntoWiki [10], which is built upon Powl, employs web 2.0 techniques to bring more features in authoring, editing, and searching semantic contents. Unlike Powl and OntoWiki which concentrate more on ontology editing, Semantic Wikipedia [158], as an extension to the popular MediaWiki software, brings semantics to texts and links in its wiki system. These systems usually have better user interfaces than those discussed before.

There exist more other semantic wiki systems (e.g., WikSAR [44], COW [58], WikiFactory [87] and etc), which provide more features such as integration with the desktop [44], typical query mechanisms [58], domain-oriented design [87], and etc.

One major issue of the most current wiki and semantic wiki systems is the reliability of the contents and the security of the systems. Wiki systems assume collaborators often contribute reliable contents. If someone makes some mistakes, others will correct it. Version control systems are used to track all the changes. Semantic wiki systems mostly rely on this assumption as well. This assumption makes wiki systems very vulnerable to malicious editing from some ill-disposed users. Human users in such cases can judge the trustability of the contents by their own existing knowledge. Computers on the other hand, have little ability to filter our incorrect knowledge created by malicious users in semantic wiki systems if they haven't been well designed. This issue has limited the use of those wiki and semantic wiki systems in some serious situations.

Since the wiki way has been proven to be an effective method to help create ontology data, this study also uses it to design the system that facilitates the authoring of ontology data for ordinary users (Chapter 4). To address the issue related to reliability and security in the conventional (semantic) wiki systems, an ontology enhanced model is proposed (Chapter 5) so that the wiki-like system that is implemented with the model can be used in some serious situations that require certain security levels.

2.5 Ontology-based Data Integration and Matching

After the discussion of the related work on the construction/development of ontology data and the generation of ontology data, we proceed to discuss the related work on ontology-based or ontology-involved data integration and matching.

The research scope of ontology-based or ontology-involved data integration and matching could be broader than web ontology data integration and matching. It includes any forms of integration of any types of data (not limited to web ontology data) as long as the ontology-related technologies have been used. This broad research area can hardly be studied from one single angle. The research perspectives of it can at least be viewed from the following three angles:

- perspective from data semantics and representations (how ontology to be used in the integration);
- perspective from data distribution and management (how to manage and organize data from different locations) and;
- perspective from data integration and match approaches (what methods to be used in integrating and matching data).

We will first overview the first two perspectives briefly and then focus on the third perspective, as the third issue that we try to tackle just falls in this perspective.

2.5.1 Data Semantics and Representations

This perspective has actually been discussed to some degree in the previous sections. For example, design of appropriate ontology languages is the first issue towards ontology-based data integration from this perspective and it has been discussed in Section 2.2.3. The discussion in this section will be focused on how ontologies are employed to organize integrated data.

Like the variety of ontology representation methods (which result in different ontology languages), the approaches to use ontologies vary. Roughly, they can be identified as following three ways: single ontology approaches, multiple ontologies approaches and hybrid approaches [160, 147]. Single ontology approaches use one global ontology (schema) to provide a shared vocabulary for the specification of the semantics exploited by multiple data sources. In multiple ontology approaches, each data source is described by its own ontology (schema) and inter-ontology mappings are created to make effective data exchange possible. Hybrid approaches are similar to multiple ontology approaches as the semantics of each source is described by its own ontology, but instead of using inter-ontology mappings, one global shared vocabulary is used for effective schema integration.

These different approaches have been used in several frameworks/systems. For example, SIMS [7] and PEPSINT [36] use single global ontology approaches; OBSERVER [113] uses multiple ontologies approaches; COIN [61] and approach by [159] use hybrid approaches. Obviously, different approaches have their own advantages and disadvantages [160]. Single ontology approaches provide a global view by deploying a single ontology schema. This is helpful for querying the integrated information; but they may not satisfy the need of changes from data sources or from ontologies themselves. Multiple ontologies approaches and hybrid approaches seem to be more flexible but this flexibility is at the expense of complexity by the introduction of inter-ontology mappings or complicated architecture.

Since this study will mostly works on the integration of ontology data within some given domains, a single global ontology schema for a given domain is often used to align data from different sources. Therefore, we will mostly use the single ontology approaches to keep the complexity at an acceptable level at the first stage. However, the extensibility to multiple ontologies approaches has also been considered for the future work.

2.5.2 Data Distribution and Management

Ontologies allow data to be presented with a more unified view to the end users. This mostly deals with the integration problem at the logical level. The distribution of the actually data in the integration can have different ways. One way is to gather all the data to a central place for integration. For example, SHOE [81] provides a language and a tool to allow users to create web pages containing semantically annotated information. Therefore such information can be crawled by a typical web robot designed to extract SHOE mark-up to form a integrated knowledge base. Similarly, Ontobroker [46] uses a component called "Ontocrawler" to extracts formal knowledge from HTML pages. Wrappers are introduced in the design of "Ontocrawler" to extract formal descriptions from large collections of web pages of similar structures. CREAM [78] is another framework focusing on relational metadata creation with component-based, ontology-driven features. Like the importance of hyperlinks over the existing web, the importance of richly interlinked, machine-understandable data over the semantic web is pointed out. Therefore, it allows to construct relational metadata (class instances and relationship instances) based on domain ontologies.

While most of ontology-based information/data integration systems require that data should be collected to a central place so that integration should proceed, recent researches on peer-to-peer network have proposed decentralized solutions to data management. Edutella [121] is a peer-to-peer infrastructure based on RDF. It uses RDF schemas that is able to represent semantics of data and provide query services that is intended to be a standardized query exchange mechanism for RDF metadata stored in distributed RDF repositories across the peers. Piazza [74] is another peer-to-peer data management system. Unlike Edutella, Piazza emphasizes on the mediation between data sources (hosted by different peers) in different formats (RDF, XML, or database). PEPSINT [36] is a framework designed to semantically integrate heterogeneous XML and RDF data sources in a peer-to-peer environment. The use of a global RDF ontology differentiates it from Edutella and Piazza.

This study has also come up with a peer-to-peer framework design. Unlike the above discussed frameworks, our framework tackles the integration issue not only from this perspective, but also from the third perspective. That is, data matching methods will be studied to improve the data quality in the peer-to-peer framework.

2.5.3 Data Integration and Match Approaches

The perspective of data integration and matching approaches has been pursued in the database research community for decades. Initially, such integration and matching approaches was studied for the data stored in databases. Research topics involves creating semantic mapping or matching between different data sources (mainly different databases). This matching problem includes schema matching and data matching. Schema matching deals with finding semantic corresponding between different database schemas; whereas data matching tries to decide if two relational tuples from two sources refer to the same real world entity [52]. Views can be regarded as a type of schema mapping between different databases. Therefore, research on using views for data integration [154, 73, 24, 75] deals with the integration problem mostly at the schema level and discusses how queries can be answered by using views. As our study assumes that a global domain ontology is used to align relevant data from different sources, the problem of schema matching has been minimized. Therefore, this study mainly focuses on data matching.

While studies on schema mapping have been extended from database schemas to ontology schemas [24], most studies on data matching are still focused on the structured data stored in databases or formatted with table-like structures. So the following will mostly discuss the data matching methods and techniques proposed for the structured data.

Overview of the Data Matching Methods

Performing data matching has been used in many applications to reduce duplications of the data. Ironically, there exist several names which refer to this very task. In the area of statistics, it is initially referred to as *"record linkages"* [125], which tries to identify duplicates when unique identifiers are not available. In some other research communities, it is also called data *cleansing/cleaning* [99], *object identification* [152], *object consolidation* [114], *entity resolution* [17], *entity reconciliation* [47], *approximate matching/joins* [68], *fuzzy matching* [6], *merge/purge* [82], *duplicate detection* [18], *deduplication* [37], and etc.

Here we show the basic definition of the data matching problem [66] that these methods try to solve. There exist several different data sources that contain data instances assumed to represent observations of entities. These instances are assumed to contain some attributes (fields or variables) identifying an individual entity. Examples of such attributes could be name, address, age and gender. Suppose source A has n_a instances and source B has n_b instances. Each of the n_b instances in source B is a potential match for each of the n_a instances in source A. So there are $n_a \times n_b$ instance pairs whose match/non-match status is to be determined. Two disjoint sets

M and U can be defined from the cross-product of A with B, the set $A \times B$. A instance pair is a member of set M if that pair represents a true match. Otherwise, it is a member of U. The data matching process attempts to classify each instance pair as belonging to either M or U.

This problem was formalized with a probabilistic model in [56] and was further extended in [172]. Basically, the model computes the ratio of probabilities of the following form:

$$R = \frac{P(\gamma \in \Gamma | (a,b) \in M)}{P(\gamma \in \Gamma | (a,b) \in U)} \tag{2.5.1}$$

where γ is an arbitrary agreement pattern or a similarity measure in a comparison space Γ between the pair of instances a and b. Given this formula, the decision rule is given as:

$$\begin{cases} R > T_\mu & (a,b) \text{ as a match} \\ T_\mu \geq R \geq T_\lambda & (a,b) \text{ as a possible match for review} \\ T_\lambda > R & (a,b) \text{ as a non-match} \end{cases} \tag{2.5.2}$$

where T_μ and T_λ are two cutoff thresholds determined by a priori error bounds on false matches and false non-matches [173].

The γ as an agreement pattern or a similarity measure is usually calculated against some field/attribute values that the instances have. Such values are generally names, addresses, and etc. To ensure effective matching, some forms of standardization (especially for names and address) are performed [173]. In addition, various similarity metrics have proposed to measure the similarity degrees between the field/attribute values of the pair. String edit distance (SED) [70] and TF-IDF [137] are two types of commonly used metrics.

Since this basic model was published, many data matching methods have been proposed, either based on this model but with more extended work, or from a new angle with a slightly different ground (for example, some machine learning methods). The following will discuss the work in categories of "unsupervised matching methods", "supervised matching methods" and "methods for relational data matching", which are highly related to our studies.

Unsupervised Matching Methods

A matching process that uses unsupervised matching methods can be either the major single process that produce the final results, or a subprocess in a multi-stage process. For example, the "blocking" method [124][123][13], often as a type of unsupervised methods, is used as a subprocess that generates candidate instance pairs for further matching. The blocking process helps create blocks of instances that have an accepted similarity degree only on a (set of) field(s)/attribute(s) such as surname or date of birth. This can reduce the potential pairs from $|A \times B|$ to a smaller number for more efficient processing.

Besides blocking methods [12], clustering methods are another type of commonly used unsupervised methods in data matching either as major single process or as subprocess. In [109], multi-stage clustering-based methods are proposed to identify duplicates in publication references. Their approach performs quick canopy clustering with two thresholds at the first stage and perform more expensive clustering methods within each canopy cluster at the second stage for refined results. In the first stage, cheap distance metrics are used in combination of simple clustering methods to speed up the creation of canopy clusters. While in the second stage, a variety of clustering approaches, such as Greedy Agglomerative Clustering (GAC), K-means and Expectation-Maximization (EM) [116][77] can be performed within each canopy cluster to refine the results using more expensive metrics and computations.

Supervised Matching Methods

In addition to unsupervised methods, supervised learning methods have also been employed to make data matching or duplicate detection more adaptive with given data. Cohen et al [33] propose an adaptive clustering method and introduces the notion of pairing function that can be learned to check duplicates. Tejada et al [153] use a mapping-rule learner consisting of a committee of decision tree classifiers and a transformation weight learner to help create mapping between records from

[12]Generally speaking, blocking methods could also be regarded as clustering method

different data sources. Bilenko and Mooney [18] use a stochastic model and Support Vector Machine (SVM) [157] to learn string similarity measures from samples so that accuracy can be improved for the given situation. These methods are more adaptive and accurate because of their various learning processes which require an adequate amount of labeled data.

It should be noted that the data instances or records for de-duplication in those mentioned methods (whether supervised or unsupervised) are not relational, which means each record is a separate instance with no explicit relation with another.

Relational Data Matching

Recently, the relations between data records have been noticed in duplicate detection research community. Singla and Domingos [140] build a collective model that relies on Conditional Random Fields (CRFs) [98][150] to capture the relation of records for de-duplication. The relationship is indicated just by common field values of data records. The model proposed by Culotta and McCallum [37], which is based on CRFs as well, deals with multi-type data records with relations other than common field values. These methods improve the accuracy of de-duplication by capturing relational features in their models. They also belong to the learning paradigm, which requires labeled samples for training the model parameters. Due to the complexity of the model, the training and inferencing require considerable time, which poses scalability problems.

To this point, we have seen that almost all the work on data matching/duplicate detection deals with structured data. While those methods can be used for ontology data matching in a straight forward way, we argue that the features of ontology data could allow us to develop even more effective methods based on them.

2.6 Summary

This chapter provides some background knowledge about ontology and its related topics. It explains what ontology is and what roles it plays in various research fields and how important the roles are.

Furthermore, an extensive (if not comprehensive) literature review has been made in the related research fields on ontology generation (including ontology schema generation and ontology data generation) and ontology-based data integration. Particularly, some most related research topics within these fields have been intensively discussed. This includes related work on ontology schema learning from the web, semantic wikis for ontology data generation and data matching methods for data integration. Their limitations and possible improvements are analyzed as well. Therefore, the next chapters will propose some new methods and techniques to address these issues.

Chapter 3

Ontology Schema Generation by Key Information Mining from the Web

3.1 Introduction

According to the previous discussion in Section 2.4.2, ontology schema generation can be assisted by ontology learning from different data sources, among which data from the web (web pages, etc) form a very important type of sources. The web has already become a successful platform on which people disseminate and collect various kinds of information. Classification, clustering, and some other data mining methods, have been applied to mine web information so that useful knowledge can be discovered for web users [25] [76]. Obviously, such discovered knowledge can also help domain ontology schema generation.

However, unlike the data stored in databases, the information in the form of web pages spread over the web, is semi-structured, heterogeneous and of huge amount. The normal data mining methods, which may work well for traditional structured data, may not be effective when they are directly applied to web pages. "Noisy information" contained in web pages, such as advertisements, copyright statements, and, etc, affects the functioning of normal data mining methods. Despite this, it has been found that using "key information" embedded in web pages can help data mining tasks such as classification [102] [175]. *Key information* refers to the certain information embedded in web pages that can categorize related web pages into several meaningful classes or identify a particular web page within a web site. For example, a distinctive menu item in each web page indicates the category of the main content in this page; a hierarchical navigation indicator shows the main topic of the page. Such menu items and navigation indicators are considered as key information as they can effectively categorize web pages into different classes. As such information is able to categorize web pages with a certain structure, it is often designed carefully in advance by the expert in the related domain. Therefore, it would help develop ontology schemas in this domain.

A straightforward method to obtain embedded key information from web pages is to extract it based on some prior observations. It is fortunate that most web pages from the same web site often have a consistent layout. By observing the pattern of the key information and its context, one can build up an information extracting tool (often called wrapper [129][8]) to extract it for a given web site. However, as the layouts of web pages and the patterns of key information always differ from site to site, a good wrapper for one web site may generate totally a wrong result for other web sites. Thus, when some web pages from a new target web site need to be processed, a new wrapper should be created by prior observations of the layout of this web site. In addition, manually building a wrapper is often time-consuming. Although there exists some wrapper induction systems that can create wrapper automatically using some machine learning techniques [118] [97] [59], most of them require several labeled web pages as examples so that inductive learning can be performed to automatically construct the wrappers. More over, such wrapper tools or systems are usually designed to extract structured data from semi-structured resources such as HTML tables in web pages. Obviously, the key information we want to extract from web pages doesn't look like this type of structured data.

This study therefore proposes a new method called Key Information Method (KIM) to help extract key information effectively. This method does not require prior observations of certain web sites for creating certain wrappers. It generates a set of candidate key information and then employs entropy evaluation to identify right key information. This study has also developed a prototype based on the KIM method and we will show its application in helping build domain ontology schema.

For the rest of the chapter, Section 3.2 discusses the common features of web pages which are related to key information. Section 3.3 presents the KIM method in detail. The experimental results about the KIM method are shown in Section 3.4. Section 3.5 shows how to use the KIM method to assist the generation of domain ontology schemas. Section 3.6 summarizes this chapter.

3.2 Key Information and Common Features of Web Pages

As we have mentioned, key information refers to the certain information embedded in web pages that can categorize related web pages into several meaningful classes or identify a particular web page within a web site. However, due to the flexible use of the HTML tags in web pages, there is no guarantee that which tag or combination of tags promises the appearance of the wanted key information. For example, the tag <H1> indicates the text enclosed by it would be a level one heading. However, some of the commercial web pages don't use it as they have their own schemes to express headings. More over, the tag <TABLE> is supposed to be used to arrange a group of structured data, but in practise, it is also widely used as a mechanism of layout design. Therefore, it is not practical to only rely on the types of tags to extract the wanted key information.

The types of HTML tags are not reliable for automatically mining key information. To handle this issue, we have conducted an observation-based survey for a considerable amount of web pages. The results show that well designed web pages often have the following features:

- **Use of information block**. Web information is grouped into blocks based on its content to make users ease of use. Within one block, the content is logically related.

- **Use of menus and navigation indicators**. Menus and navigation indicators (also called "breadcrumb navigation") are widely used in web sites. Automatic recognition and reuse of them would be beneficial for data management tasks.
- **Similarity of word length of items in menus/navigation indicators**. Though the schemes of menus (the tags used, the styles rendered) differ from site to site, two features are common in most web sites: word length of the menu items in a web site is same or similar; and word length of each item is quite short (around two or three words). We believe the selection of words for menus is always well considered by the web site constructors so that the resulted menu is informative in content but concise and balanced in its appearance.

Figure 3.1 presents four examples of menus/navigation indicators from different web sites [1]. It is easy to see their styles differ, but the word length of each item is similar and less than three words. Such common features can be used to develop general and efficient methods to extract key information from a large amount of web pages.

3.3 A Key Information Mining Method

3.3.1 Overview of the Method

The KIM method consists of two main steps as illustrated in Figure 3.2:

Step 1: Generation of candidate key information: to extract a candidate key information list (CKIL) from each web page due to a set of given criteria; to merge obtained CLKILs to form a site candidate key information list (SCKIL);

[1]obtained from http://www.webdesignpractices.com/navigation/breadcrumb.html and http://www.pcmag.com

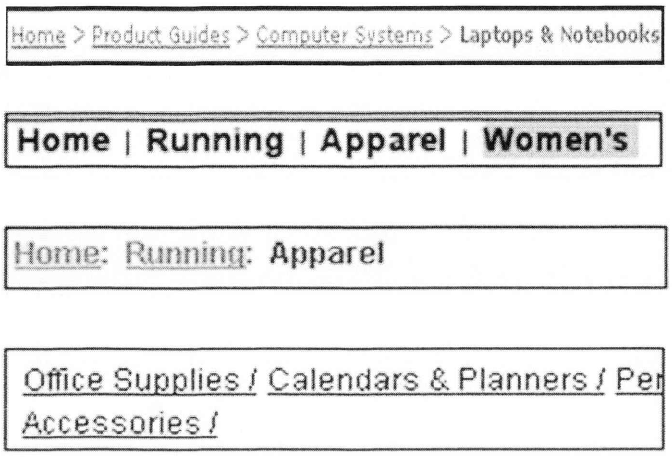

Home > Product Guides > Computer Systems > Laptops & Notebooks
Home | Running | Apparel | Women's
Home: Running: Apparel
Office Supplies / Calendars & Planners / Pe
Accessories /

Figure 3.1: Word length in menus/navigation indicators of different web sites

Step 2: Entropy evaluation: to apply the entropy evaluation technique to SCKIL so that real key information can be identified.

The following two subsections will give detailed explanations of the two main steps.

3.3.2 Generation of Candidate Key Information

This subsection presents the technique to generate candidate key information (CKIL and SCKIL) from web pages. To make the description for this step clearer, we first define some concepts as follows:

Definition 3.1 (text node). We regard "CDATA" embedded in the tags of a web page DOM tree as text node with the exception of those embedded in the tag <SCRIPT>, <STYLE> and <SELECT>, as texts in them are just functional codes, style or form data. An <IMG> tag is also treated as a text node if its attribute "alt" is set with values.

Definition 3.2 (word length). The word length of a text node is the number of words the node contains. The words here do not include separate characters (i.e. space, table character) or other meaningless characters. If the word length equals zero, the corresponding node will be ignored.

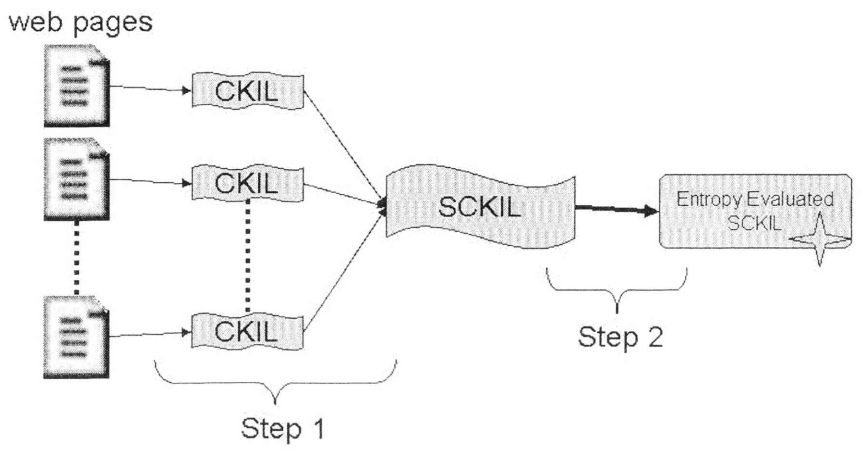


Figure 3.2: Overview of the KIM method

Definition 3.3 (menu subtree)**.** A menu subtree is a subtree of the DOM tree of a web page, which contains only text nodes whose word length is less than α. The value of α can be adjusted. We commonly let α=5. Choosing α as five is based on our observations on a large amount of sample web pages that the word length of a menu item is often less than 5.

Definition 3.4 (menu item)**.** Menu items are created from a menu subtree. A menu item I has two components, denoted by (*name*, *style*), where *name* is the text extracted from a text node of a menu subtree and *style* is the style pattern created by concatenating the tag information from the root of the menu subtree down to the leave of the text node. The tag information includes the name of the tag and its attribute lists.

Definition 3.5 (menu instance)**.** A menu instance MI is denoted as ($IList$, $Pages$), where $IList$ is a list of menu items created from one specific menu subtree; and $Pages$ is a set of web pages that contain this menu subtree. An initially created MI from one page only has this page in its set of $Pages$. If $MI.Pages$ contain a page p, then p is called to be associated with this MI.

Definition 3.6 (similarity of menu instances)**.** Two menu instances MI_i and MI_j extracted from two different web pages are compared according to their first menu item (i.e., $IList[0]$). MI_i and MI_j are **similar** (denoted by $MI_i \approx MI_j$) if and only if:

$$
\begin{gathered}
MI_i.IList[0].name = MI_j.IList[0].name \\
or \\
MI_i.IList[0].style = MI_j.IList[0].style \\
\hfill i \neq j, \quad 0 \leq i, j < N
\end{gathered}
$$

where N is the number of web pages. It should be noted that two similar menu instances may not be identical. MI_i and MI_j are identical (denoted by $MI_i = MI_j$) only if:

$$
\begin{gathered}
MI_i.IList[0].name = MI_j.IList[0].name \\
\textbf{and} \\
MI_i.IList[0].style = MI_j.IList[0].style \\
\hfill i \neq j, \quad 0 \leq i, j < N
\end{gathered}
$$

If $MI_i = MI_j$, then they can merge into one menu instance MI with $MI.IList = MI_i.IList$ and $MI.Pages = MI_i.Pages \cap MI_j.Pages$.

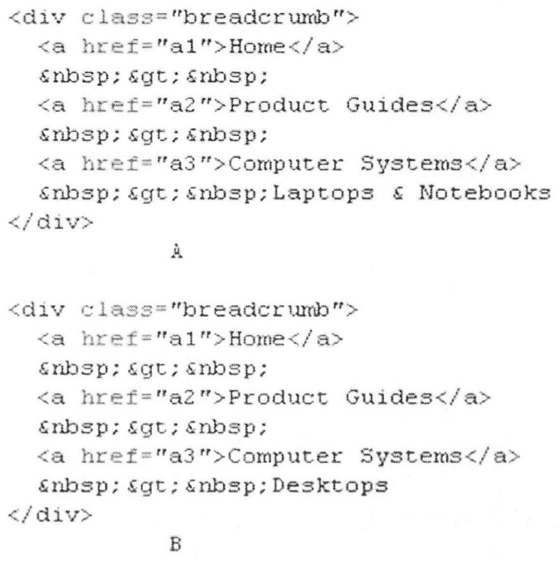
Definition 3.7 (site menu). A site menu M consists of a set of similar menu instances extracted from different web pages collected from one web site, denoted by $MISet$. Thus, $\forall MI_i, MI_j \in MISet(i \neq j)$, we have $MI_i \approx MI_j$. However, if $MI_i = MI_j$, they should merge into one menu instance.

By these definitions, we can notice that the candidate key information is able to be expressed in a more concrete concept "menu". A CKIL is denoted as a menu instance list of a certain web page, and a SCKIL is corresponding to the site menu list.

It should be noted that the definition of the menu subtree is so loose that some other parts of the web pages may become menu instances. However, as far as the menus containing key information can be extracted, the definition is acceptable, as we will perform an entropy evaluation later to discover them.

Figure 3.3 is an example (adopted from http://www.pcmag.com) to explain these definitions. A and B are menu subtrees, as they only contain text nodes with word length less than 5. The first menu item of menu instance created from A has name "Home" and style "<div class=breadcrumb><a href=a1>". The two corresponding menu instances are similar because their first menu item is the same.

However, they are not identical due to the difference of their fourth menu item (i.e. "`&nbsp;&gt;&nbsp;Laptops & Notebooks`" from A and "`&nbsp;&gt;&nbsp; Desktops`" from B).



```
<div class="breadcrumb">
  <a href="a1">Home</a>
  &nbsp;&gt;&nbsp;
  <a href="a2">Product Guides</a>
  &nbsp;&gt;&nbsp;
  <a href="a3">Computer Systems</a>
  &nbsp;&gt;&nbsp;Laptops & Notebooks
</div>
```

A

```
<div class="breadcrumb">
  <a href="a1">Home</a>
  &nbsp;&gt;&nbsp;
  <a href="a2">Product Guides</a>
  &nbsp;&gt;&nbsp;
  <a href="a3">Computer Systems</a>
  &nbsp;&gt;&nbsp;Desktops
</div>
```

B

Figure 3.3: Two illustrative menu subtrees

Using the above definitions 3.1 to 3.7, we propose an algorithm (shown in Figure 3.4) that can extract a list of menu instances from a given web page. This algorithm is a bottom-up procedure with recursive design. It first checks the leaf nodes to mark if they are valid text nodes or not. The parameter α, given a value "5" for example, has been discussed in the definition of menu subtree. The up-level nodes are then examined. According to the status of marks their children have, the algorithm will determine if menu instances are created or not. The parameter β, usually given a value "1", is used to ensure each menu instance to have at least one menu item. All the menu instances are extracted and added into a menu instance list (MIL) finally.

After a menu instance list is extracted from each web page, all these menu instance lists from the same web site can be merged into a one-site menu list. The merging process is controlled by the definition of similarity of menu instances and

```
Input: n: root node of one DOM tree
Output: MIL: menu instances list

getMIL(n) {
  if (n is a text node){
    l = word length of n;
    if (l>0 and l<$\alpha$)
      mark n as Y;
    else if (l>=$\alpha$)
      mark n as N;
  }else if (n has child) {
    for (each child i of n)
      getMIL(i);
    if (NO child of n marked N){
      CountY = numbers of mark Y;
      if (CountY >=$\beta$) {
        regard n as menu subtree;
        create menu instance MI from n;
        add MI to MIL;
        mark n as N;
      }
    } else {// some children marked N
      for (each child i of n not marked N){
        regard i as menu subtree;
        create menu instance MI from i;
        add MI to MIL;
        mark i as N;
      }
      mark n as N;
    }
  }
}//end of getMIL
```

Figure 3.4: An algorithm for extracting a menu instance list from a web page

site menus, as discussed before. After the merge, a site menu list is generated for entropy evaluation.

3.3.3 Entropy Evaluation

As the definition of the menu subtree is quite loose, the list of site menu may contain a lot of "false menus". Therefore, an evaluation mechanism is needed to find out the real informative menus that contain key information.

There are three main kinds of site menus. The first kind is called "noisy information menus". Advertisements and copyright information can be treated as this kind of menus. Such noisy information always constantly appears in web pages across the web site. This kind of information, if treated as menus, may only have one menu instance with many web pages associated. For example, there are 100 web pages from one web site X. A site menu containing an advertisement may have only one menu instance associated with 95 pages. Obviously, this kind of menus is "false menus". The second kind of menus is called "accidental menus", as some parts of the main contents of the web pages sometimes can be accidentally treated as menus according to the loose definition of a menu subtree. In this case the menu instances of these menus usually have little web pages associated. Continue the example of the site X: an "accidental menu" may contain two instances: one is associated with two web pages and the other with three web pages. This kind of menus is "false menus" too. The third kind of menus is those menus containing real key information. Such menus always have several instances, each of which are associated with a number of web pages. For the example site X, a menu containing key information may have four instances associated with 10, 15, 20 and 30 web pages respectively. We regard a menu as a random variable with several values (instances) it can take. If the probability of taking each value is closer, the entropy of the random variable would be greater. Therefore, an entropy evaluation is suitable for distinguishing menus containing key information.

Definition 3.8 (entropy of menus). Suppose a menu M has n menu instances and the i-th instance has p_i web pages associated. The total number of web pages from the web site is S. Let $N = \sum_{i=1}^{n} p_i$, then $S \geq N$. The entropy of menu M can be calculated using the following formula [35] according to the definition of entropy in information theory:

$$H(M) = \begin{cases} -\sum_{i=1}^{n} \frac{p_i}{S} \log \frac{p_i}{S} & S = N \\ -\sum_{i=1}^{n} \frac{p_i}{S} \log \frac{p_i}{S} - \frac{S-N}{S} \log \frac{S-N}{S} & S > N \end{cases} \tag{3.3.1}$$

Where $S > N$ means that some web pages are not associated to any instances of the menu M. In this case, we assume there is a "fictional instance" with $S - N$ web pages associated.

In order to compare the entropies of different menus, we normalize them to a fixed range of values. Given the property of the entropy: $0 \leq H(X) \leq \log r$, where

r denotes the number of all possible values that random variable X can take, we calculate the *normalized entropy* for each menu using the following formula [35]:

$$NH(M) = \begin{cases} \frac{H(M)}{\log n} & S = N \\ \\ \frac{H(M)}{\log(n+1)} & S > N \end{cases} \tag{3.3.2}$$

Continue using the above example of the web site X, we calculate normalized entropies of these three kinds of menus. The normalized entropy of the noisy menu that contains advertisement is 0.29. For the accidental menu, its normalized entropy is 0.21. The real key information menu has normalized entropy value 0.96. Therefore, a menu with higher normalized entropy is more likely to contain key information that we expected.

A set of experiments have shown that the above evaluation method works well. However, sometimes it may be fooled by some "false menus" which have only one instance associated with around half of all web pages from a related web site. These "false menus" are usually some fixed advertisements existing on a proportion of the web pages. To overcome this problem, we also calculate the normalized entropy of the menus without considering the "fiction instances" (denoted by NHx). The final value used to evaluate the site menu is the *average normalized entropy* (ANH) of NH and NHx.

$$NHx(M) = \frac{-\sum_{i=1}^{n} \frac{p_i}{N} \log \frac{p_i}{N}}{\log n} \tag{3.3.3}$$

$$ANH(M) = \frac{NH(M) + NHx(M)}{2} \tag{3.3.4}$$

After ANH of each site menu is calculated, they are resorted according to this value. A new list of a site menu is presented so that its users can pick out those that contain key information by checking a few top ranked menus.

3.4 Experiments

This section demonstrates the experimental results of the proposed KIM method. To conduct the experiment, we have collected the web pages from five well known web sites for this experiment: ABC news [2], Yahoo news [3], CNET [4], PC Magazine [5], and PC World [6]. The method used to collect web pages is a breadth first crawling within a given web site. Due to the large size of web pages one site can have, the collected web pages are only a proportion of all the pages from each web site.

We have applied the KIM method to the web pages from each site to obtain an entropy sorted list of site menus. From this list, key information of the web pages can be discovered. Table 3.1 presents the experiment results for the five selected web sites. The column "Number of Pages" denotes the number of web pages collected from the corresponding web sites. "List Length" means the length of list containing extracted site menus. Due to the looseness of the site menu extraction method and the noisy nature of the commercial web pages, this list is normally quite long. The average normalized entropy values of the key information are listed in Column 4. "Rank" is the rank position of the key information in the candidate list according to the average normalized entropy values. For the web site ABC news and Yahoo news, this key information is the categories of the news (e.g. politics, technology, or etc); For CNET, PC Magazine, PC World, it is the topics of the review articles, purchase guides (e.g. digital cameras, laptops, printers, or etc). It is obvious that the rank position of the key information is quite up with a high average normalized entropy value. For example, key information in the web site ABC news ranks the first (as shown in table), which means it has the highest average normalized entropy in the candidate list. Key information in the web site PC Magazine ranks the second (also as shown in Figure 3.5). A little of "false" key information still has high values

[2]http://abcnews.go.com
[3]http://news.yahoo.com
[4]http://reviews.cnet.com
[5]http://www.pcmag.com
[6]http://www.pcworld.com

Table 3.1: Experiment results for five web sites

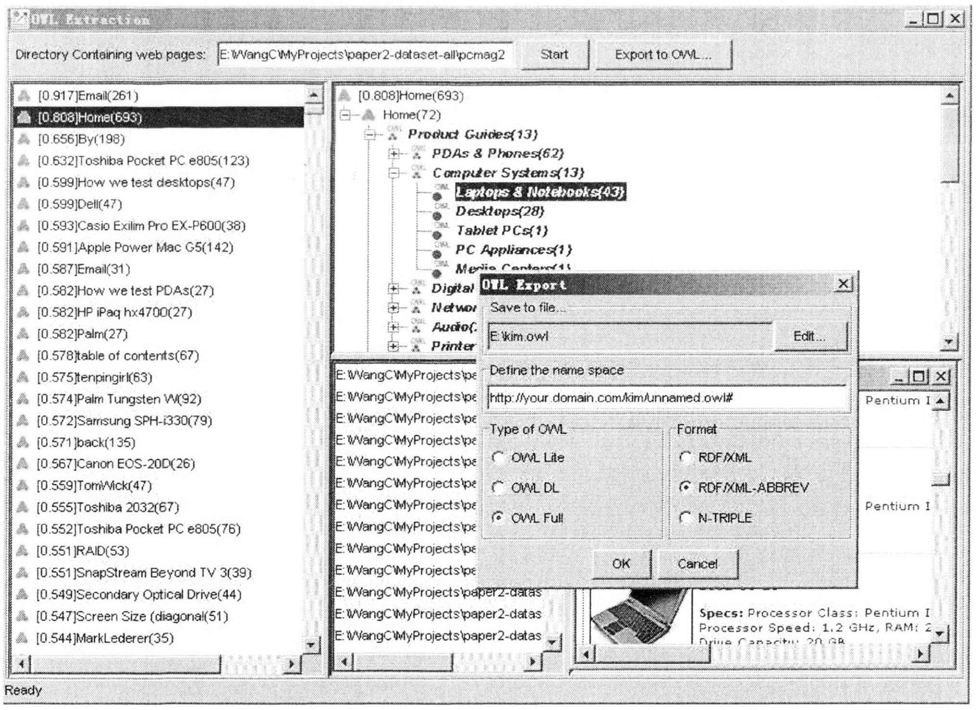
Web site	Number of pages	List length	Average normalized entropy	Rank
ABC news	671	484	0.929	1
Yahoo news	492	558	0.924	2
CNET	786	5837	0.666	3
PC magazine	700	1235	0.808	2
PC world	2360	847	0.877	1

for some web sites, for instance, the CNET web site. This is because the web pages from these web sites contain lots of small sentences or short phrases. These noises are all treated as potential key information, building up a very long site menu list (for CNET, 5837 long). Thus the effectiveness of the entropy evaluation is slightly degraded. However, the results are acceptable since the rank is still high compared with the long length of the candidate key feature list.

3.5 Domain Ontology Schema Generation with KIM

Key information can contribute to certain domain ontology generation. According to the research results of Uschold and Gruninger [155], ontology generation or development consists of three main phases. Building the ontology which includes ontology capture, ontology coding and integrating existing ontology is the second phase and the most important one. From this view, using key information to help domain ontology generation is able to help accelerate this phase as the taxonomy knowledge captured by key information can alleviate the work of domain experts.

We have developed a prototype based on the KIM method and have applied it to help domain ontology generation. First, when the scope of domain ontology has been identified, users are expected to provide seed URLs of web sites related to the ontology. This process can be assisted by using major search engines such as Google (http://www.google.com) and Yahoo (http://www.yahoo.com) or online directories such as Google Directory (http://directory.google.com) and Open Directory Project (http://dmoz.org). Once the related URLs are collected, crawling is performed to gather web pages from those web sites. These web pages are then fed into our application and key information mining is conducted. The mining results are presented to the users via a friendly user interface so that they can interact with the application to check the results and to select the wanted information items for ontology generation. Those selected items can then be exported into an ontology file with OWL as description language. As mentioned before, OWL [111] is a language developed by W3C to describe web ontologies and is widely used in the semantic web research community. Finally the exported ontology file can be imported into Protégé [60], the popular ontology editing tool. The users are then able to modify the raw ontology in the Protégé environment without having to build a new ontology from the scratch.



```
Directory Containing web pages: E:\WangC\MyProjects\paper2-dataset-all\pcmag2

[0.917]Email(261)
[0.808]Home(693)
[0.656]By(198)
[0.632]Toshiba Pocket PC e805(123)
[0.599]How we test desktops(47)
[0.599]Dell(47)
[0.593]Casio Exilim Pro EX-P600(38)
[0.591]Apple Power Mac G5(142)
[0.587]Email(31)
[0.582]How we test PDAs(27)
[0.582]HP iPaq hx4700(27)
[0.582]Palm(27)
[0.578]table of contents(67)
[0.575]tenpingirl(63)
[0.574]Palm Tungsten W(92)
[0.572]Samsung SPH-i330(79)
[0.571]back(135)
[0.567]Canon EOS-20D(26)
[0.559]TomWick(47)
[0.555]Toshiba 2032(67)
[0.552]Toshiba Pocket PC e805(76)
[0.551]RAID(53)
[0.551]SnapStream Beyond TV 3(39)
[0.549]Secondary Optical Drive(44)
[0.547]Screen Size (diagonal(51)
[0.544]MarkLederer(35)

[0.808]Home(693)
  Home(72)
    Product Guides(13)
      PDAs & Phones(62)
      Computer Systems(13)
        Laptops & Notebooks(43)
        Desktops(28)
        Tablet PCs(1)
        PC Appliances(1)
        Media Centers(1)
      Digital
      Networ
      Audio(
      Printer

Save to file...
E:\kim.owl
Define the name space
http://your.domain.com/kim/unnamed.owl#
Type of OWL          Format
( ) OWL Lite         ( ) RDF/XML
( ) OWL DL           (•) RDF/XML-ABBREV
(•) OWL Full         ( ) N-TRIPLE

Specs: Processor Class: Pentium I
Processor Speed: 1.2 GHz, RAM: 2
```

Figure 3.5: The interface of key information mining and OWL class marking & exportation

Figure 3.5 illustrates the interface of the developed application prototype. For example, to develop an ontology relating to the domain of electronic products, we first identify web sites that publish information about this domain. Web pages are then crawled from the identified web sites. These web pages are analyzed by the KIM method and the result (the entropy ranked list of key information) is derived (shown in the left area of Figure 3.5). The hierarchical structure of the extracted information items is displayed in the top right area and the web pages relating to the selected information item are listed in the bottom right area. After marking the desired items as concepts relating to the target domain of electronic products, users can export them into an OWL format file, which can be opened in Protégé for further editing and refinement (Figure 3.6).

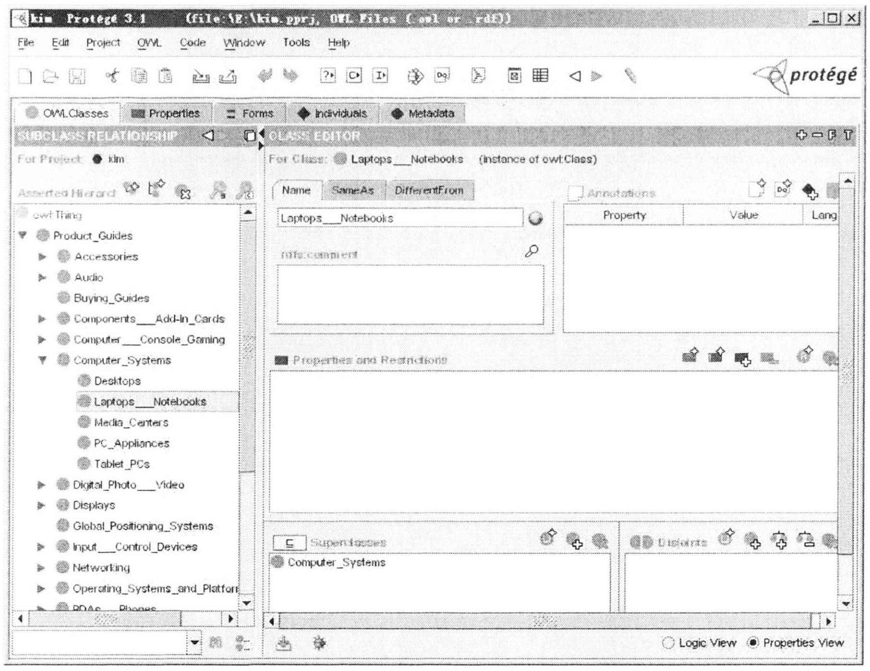


Figure 3.6: The exported raw owl file opened in Protégé for further editing and refinement

It is easy to see that this prototype application enables users to find the key information easily for ontology schema generation in some interested domains. However, it should be noted that this method and prototype may not work well when the domains don't have abundant and well-organized resources on the web.

3.6 Summary

This chapter points out the existence of key information in web pages and the use of key information for developing domain ontology schemas. It has also proposed KIM, a two-step method that can automatically mine key information from web pages. The method first extracts a list of candidate key information. It then uses an entropy evaluation to filter most of the noisy information in the list so that the key information can be discovered easily. Our experimental results show that it is effective in mining key information from web pages. Based on the KIM method, an application prototype has been developed to make use of the extracted key information for domain ontology schema generation. The intended users can mark the desired information items and export them as ontology concepts together with their hierarchical relations. By using ontology editing tools, the users can edit and refine those concepts easily. Therefore, it helps alleviate the work of developing the domain ontology schema from the scratch and reduce the involvement of domain experts in certain cases.

Chapter 4

Web Ontology Data Generation

4.1 Introduction

While last Chapter has shown a web mining method that can be used for the generation of domain ontology schemas, this chapter will focus on the topics of generation and management of web ontology data.

Generation of web ontology data is as important as the development of ontology schema itself in terms of the vision of the semantic web. Use the current web as an example. The abundance of information and data in the current web have made it an important place for people to seek information. In the same sense, the abundance of ontology data in the semantic web will be also very crucial. However, through the semantic web search engine Swoogle [49], we've found that ontology data does not seem to be very abundant on the web in contrast with the amount of ontology schemas.

As discussed in Section 2.4.3, there are several types of methods for generating ontology data, including: (1) Creating new ontology data using some sophisticated ontology development editors or tools; (2) Mapping existing structured data using views; (3) Annotating existing data (mostly web pages) using some tools or with certain automatical techniques. And (4) Creating new ontology data by extending wiki-like collaborating systems. However, in a dynamic and open environment like the web or the semantic web, not all these methods can be used with good efficiency or effectiveness. This study therefore proposes two methods, trying to make the generation of web ontology data easier and more efficient. The first method is designed to deal with existing data which belong to a certain domain but have not yet been interpreted by certain ontology schemas. It goes beyond the mapping methods by converting such structured or semi-structured data into web ontology data. The use of the XQuery for conversion in the method makes the process much more flexible. The other method is designed to deal with the creation of new ontology data from ordinary users (typically web users). Following this method, a web-based system *robinet* is developed so that users are enabled to author new ontology data collaboratively in a wiki-like style. But unlike other wiki-like approaches, the system can be extended with more management functions based on the underlying ontologies.

For the rest of the chapter, Section 4.2 presents the method that deals with the generation of web ontology data from existing data by conversion, and also has a discussion on the data formats related to web ontology data. Section 4.3 shows the prototype system that allows ordinary users to author new ontology data according to given ontology schemas, including the overview of the system and its main functions. Section 4.4 discusses features and limitations of these methods. Section 4.5 summarizes this chapter.

4.2 Web Ontology Data Conversion

4.2.1 Data Formats

Due to different applications or circumstances, existing data have a variety of formats. We mainly focus on structured or semi-structured data, which include data that are well modeled and stored in database, XML data, RDF/RDFS metadata, ontology data and other well formatted data (e.g. bibtex). For data like those written in HTML files, special processes such as information extraction are required to transform them into structured data.

We briefly discuss these different data formats and their relations even though some of them may have been mentioned before. Firstly, structured data stored and managed in different database management systems (DBMS) (e.g. Oracle, SQL Server, DB2, MySQL and etc) are mostly modeled based on relational modeling theories. Different DBMS have different physical solutions to store such data, but at the logical level, they use tables, views, fields and etc to present data with Structured Query Language (SQL) used for data structure definition and data manipulation. The use of SQL helps data to be transformed from different databases (but mostly limited to databases managed by the same DBMS). Secondly, as mentioned earlier, XML [20] is a simple, very flexible text format, which was originally designed to meet the challenges of large-scale electronic publishing. Now, XML is also playing an increasingly important role in the exchange of a wide variety of data on the web, between different databases (not limited to databases managed by the same DBMS) and elsewhere. Compared to HTML, XML provides more sophisticated functions that allow user to design their own tags to express structures and particular meanings of the data. These functions enable users to export data stored in database into XML format with customized tags. Thirdly, Resource Description Framework (RDF) [108] is a framework that provides a model to describe online resources using meta data. The contents described with this framework are often serialized using XML syntax. That is, a RDF file is often a XML file as well. In addition, RDF Schema (RDFS) [22] defines the vocabulary used in RDF data models, making itself suitable for basic ontology schema construction. Finally, OWL [111], as discussed before, is a family of languages especially designed for describing web ontologies. It can also be serialized using XML syntax.

From above discussion, it is easy to see that XML has become a "universal" language which helps to exchange data between databases. Many other languages including OWL are also based on it. However, unlike OWL which has the functionality to describe data with rich semantics, XML only encodes the data with predefined tags.

That is, data encoded in raw XML format have very little semantics. Since such raw XML data have become more popular, it is desirable to convert them to a form that has rich semantics to interpret them.

4.2.2 Using XQuery for Conversion

XQuery [19] is an XML query language which has become a W3C recommendation. Similar to SQL, XQuery is designed to be a language in which queries are concise and easily understood. However, while SQL can only be applied to structured data in databases, XQuery is flexible enough to query a broad spectrum of XML information sources, including both databases and documents, whether structured or semi-structured.

XQuery can be used to convert ordinary XML data into OWL formatted data which is aligned with intended ontology schemas. Executed by a XQuery-compatible engine, a query written in XQuery takes XML files as input and generates results in an XML format defined by the query itself. Since OWL is also based on XML syntax, the problem is then transformed into how to design the query so that the output results are actually the desired OWL format. The advantage of using XQuery instead of developing custom programs for conversion is obvious. With custom programs, it is often necessary to design custom conversion rules and to maintain the program, which might be time-consuming and error-prone. On the other hand, XQuery, as a W3C recommendation, is versatile enough to satisfy our needs for typical XML data conversion into OWL format. In addition, there has already been several implemented query engines to choose. Therefore, to convert XML data to desired OWL format, we only focus on the design of the queries and use a chosen XQuery engine to process them. In our study, *Nux* (http://dsd.lbl.gov/nux/), a java toolkit which is capable of XQuery processing, is used for such processing. Figure 4.1 illustrates a typical query used for the conversion. Figure 4.1 (a) shows the source data in XML format; Figure 4.1 (b) illustrates the query segment written in XQuery used for the conversion; And Figure 4.1 (c) shows the converted data in OWL format.

```
<?xml version="1.0" encoding="UTF-8"
   standalone="yes"?>
<pubs>
  <pub>
    <id>14</id>
    <type>JournalArticle</type>
    <author>Mike</author>
    <author>Jane</author>
    <pubDate>2005</pubDate>
    <title>
      Solution Definition ...
    </title>
    <inBook>
      IEEE Computational ...
    </inBook>
    <volume>Vol. x</volume>
    <issue>No. y</issue>
    <pageNumbers> a-b.</pageNumbers>
  </pub>
  ...
</pubs>
```

(a)

```
...
for $pub in doc("example.xml")/pubs/pub
return
  element {fn:concat("u:",$pub/type/text())} {
  attribute rdf:ID {$pub/id/text()},
  for $author in $pub/author
    return
      element u:author {
        element u:Person {
          element u:name  {$author/text()}
        }
      },
  element u:pubDate { $pub/pubDate/text()},
  element u:title {$pub/title/text()},
  element u:pp {$pub/pageNumbers/text()},
  element u:issue {$pub/issue/text()},
  element u:volume {$pub/volume/text()},
  element u:inBook {
    element u:Publication {
      element u:title {
        $pub/inBook/text()
      }
    }
  }
}
...
```

(b)

```
...
<u:JournalArticle rdf:ID="14">
  <u:author>
    <u:Person>
      <u:name>Mike</u:name>
    </u:Person>
  </u:author>
  <u:author>
    <u:Person>
      <u:name>Jane</u:name>
    </u:Person>
  </u:author>
  <u:pubDate>2005</u:pubDate>
  <u:title> Solution Definition ... </u:title>
  <u:pp> a-b.</u:pp>
  <u:issue>No. y</u:issue>
  <u:volume>Vol. x</u:volume>
  <u:inBook>
    <u:Publication>
      <u:title>IEEE Computational ...</u:title>
    </u:Publication>
  </u:inBook>
</u:JournalArticle>
...
```

(c)

Figure 4.1: An XQuery used to convert XML into OWL

For the data stored in different types of databases, XML is used as an intermediate format to export them. Most modern database management systems support the functionality of exporting data into XML format. Therefore, these data can be easily exported into XML files, which can then be transformed into OWL by applying specific XQuery queries.

4.3 Web Ontology Data Authoring

While existing data is very useful for the ontology data generation, it is equally important to allow new data to be created for ontology related applications. Since an easy and collaborative environment will encourage ordinary users to author and publish data (which has been proven by the wiki model), this study develops a web-based prototype system with such features to facilitate the authoring of web ontology data in given domains.

4.3.1 System Overview

This subsection overviews the prototype system *robinet* used for web ontology data authoring. It shows the general structure of the system and its design principles and functions. The system is implemented in Java with JSP and Servlet technology.

Figure 4.2 illustrates the general structure of the *robinet* system and the interaction between it and its managed web ontology data and users. At its lowest layer, the data layer, both the ontology schemas and generated ontology data are stored. Above the data layer is the API layer, which is made up of low level packages and the APIs they offer. To implement the function of ontology data generation, a useful package *Jena* [1] , which provides APIs to access web ontology schemas and data, is placed in this layer. Based on these APIs, a supporting layer is constructed to provide customized tools which are used to build various operations, data models and functions for presentation. The propose of this layered design is to make the implemented system easy to be maintained and extended. Particularly, at its upper level, the system is implemented with the well known model-view-controller (MVC) design pattern.

The following main principles are considered when we design the *robinet* system.

Modularity. This actually has become a common practice in software development. Guided by this principle, we try to divide the system into several loosely coupled modules and group them into layers. This will allow us to modify or update a module with little interference to other modules. Figure 4.2 actually reveals this principle.

Friendly URLs. Unlike some web sites that use very long and strange URLs for their web contents, wiki systems often use URLs that are both user and search engine friendly. This feature has become a trend nowadays. For the semantic web, this is not just a fashion, as a friendly URL is often a permanent URL that is used to identify resources on the semantic web. Therefore, the *robinet* system uses friendly URLs, particularly for concepts and instances. The following subsections will discuss

[1]Jena, http://jena.sourceforge.net

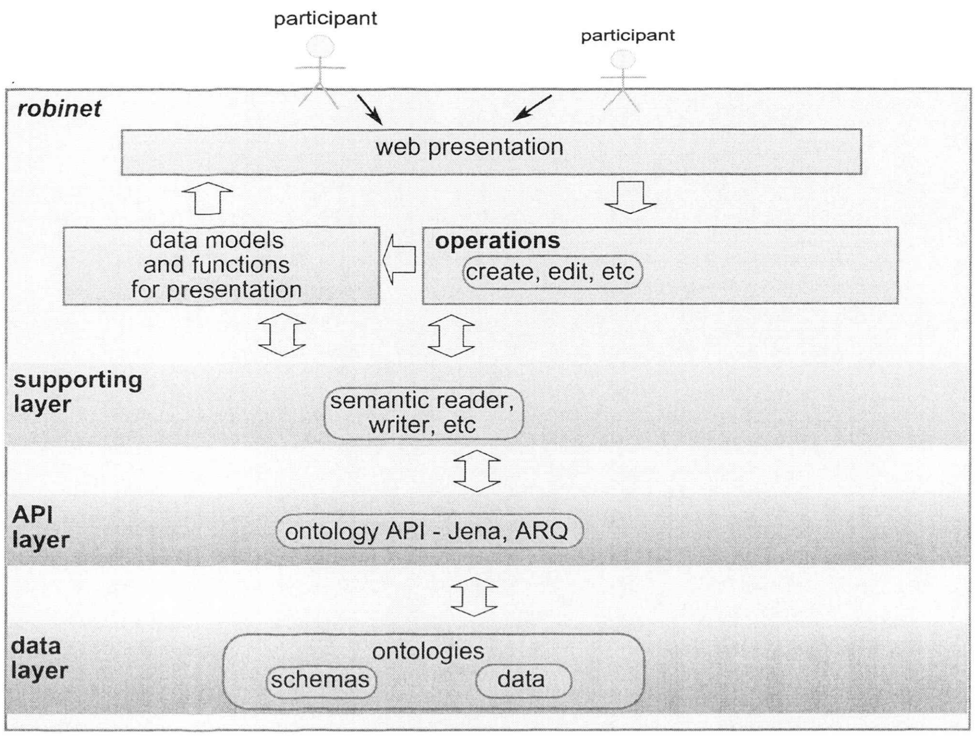


Figure 4.2: The general structure of the prototype system *robinet*

how we try to stick to this principle.

Ease of use. Our system should provide enough facilities to make ordinary users feel comfortable when they use it. The following subsections will show how easy for users to author web ontology data.

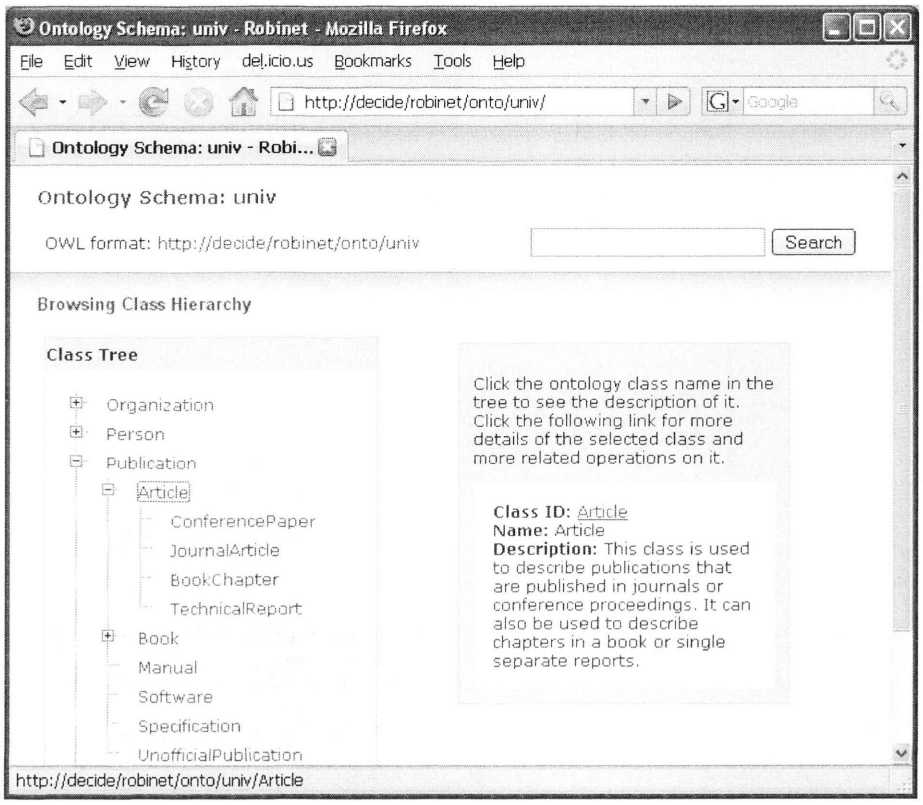
Along with the above design principles, we have designed and implemented essential functions to help web ontology data authoring. Such functions allow users to browse ontology schemas, create and edit instances and etc. The following subsections will discuss them in detail.

4.3.2 Ontology Schema Browsing

Browsing ontology schemas helps ordinary users get familiar with the domain the ontology is intended to describe. It is clear that a user without an adequate understanding of the domain ontology schema may not be able to proceed and create the correct information for the domain. As discussed before, a domain ontology schema mostly contains classes, properties, and relations between classes. When they are encoded in the popular OWL language, ordinary users may have difficulty to read and understand them intuitively as OWL is designed primarily for computers to process. Some ontology specific editing tools (e.g., Protégé [60], OntoEdit [148], SWOOP [92]) can provide graphical user interfaces that allow users to view the domain ontology schemas intuitively. But this requires users to install additional software. In addition, since such editing tools are mostly designed for domain experts or certain professionals to develop ontologies in a desktop environment, they are not web-based solutions preferred by ordinary users. Therefore, it is desirable to have a web-based browsing mechanism so that users can just use any types of browsers to learn the domain ontology schemas before they are ready to contribute information.

Initially, the *robinet* system requires the ontology schemas for given domains to be loaded for browsing. In addition, some initial settings need to be configured so that the system can function properly including allowing ontology schema browsing. These initial settings include the system's URL base, the actual directories that store ontology schemas and data, and etc. Such initial setup can easily be conducted by a system administrator.

Once the initial setup is done, users can browse the ontology schemas using common browsers such as Internet Explorer or Firefox. Figure 4.3 shows the web interface which renders the classes (concepts) in an ontology schema about the academic domain in a tree structure according to their is-a relationship. Users can get familiar with these classes and their relations by exploring this tree. In addition, if they are interested in a particular class, they can click it in the tree to see detailed comments


Ontology Schema: univ - Robinet - Mozilla Firefox
File Edit View History del.icio.us Bookmarks Tools Help
http://decide/robinet/onto/univ/
Google
Ontology Schema: univ - Robi...
Search

Ontology Schema: univ

OWL format: http://decide/robinet/onto/univ

Browsing Class Hierarchy

Class Tree

- Organization
- Person
- Publication
  - Article
    - ConferencePaper
    - JournalArticle
    - BookChapter
    - TechnicalReport
  - Book
  - Manual
  - Software
  - Specification
  - UnofficialPublication

Click the ontology class name in the tree to see the description of it. Click the following link for more details of the selected class and more related operations on it.

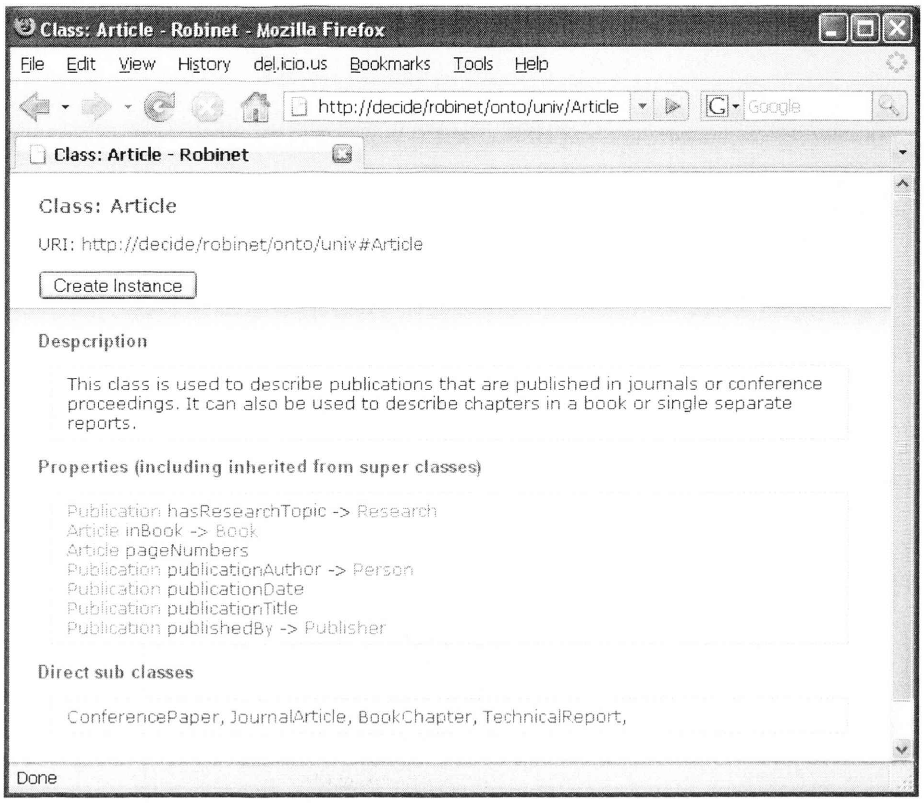
Class ID: Article
Name: Article
Description: This class is used to describe publications that are published in journals or conference proceedings. It can also be used to describe chapters in a book or single separate reports.

http://decide/robinet/onto/univ/Article

Figure 4.3: The interface that allows users to browse the structure of the domain ontology schema

(if supplied by the schema makers) on this class in the right-hand side panel. If further information is needed to know about this class, users can follow the link in the right-hand side panel to view such information. For example, Figure 4.4 shows the web page that tells about the class of `Article` in the ontology schema. This web page not only shows the class's Human friendly labels, detailed comments and actual URL/URI which can be used to identify instances for semantic web applications, but also properties associated with it and its sub classes. These properties and sub classes are linked to their corresponding web pages. So user can follow such links to learn more about them whenever necessary.

It should be noted that these web pages are all dynamically generated by the


Class: Article - Robinet - Mozilla Firefox
File Edit View History del.icio.us Bookmarks Tools Help
http://decide/robinet/onto/univ/Article
Google
Class: Article - Robinet
Create Instance
Done

Class: Article

URI: http://decide/robinet/onto/univ#Article

Despcription

This class is used to describe publications that are published in journals or conference proceedings. It can also be used to describe chapters in a book or single separate reports.

Properties (including inherited from super classes)

Publication hasResearchTopic -> Research
Article inBook -> Book
Article pageNumbers
Publication publicationAuthor -> Person
Publication publicationDate
Publication publicationTitle
Publication publishedBy -> Publisher

Direct sub classes

ConferencePaper, JournalArticle, BookChapter, TechnicalReport,

Figure 4.4: The interface that shows the details of the class `Article` in the ontology schema

system according on the given ontology schema. As far as the proper ontology schema has been developed and loaded into the system, there's no need to create these web pages manually.

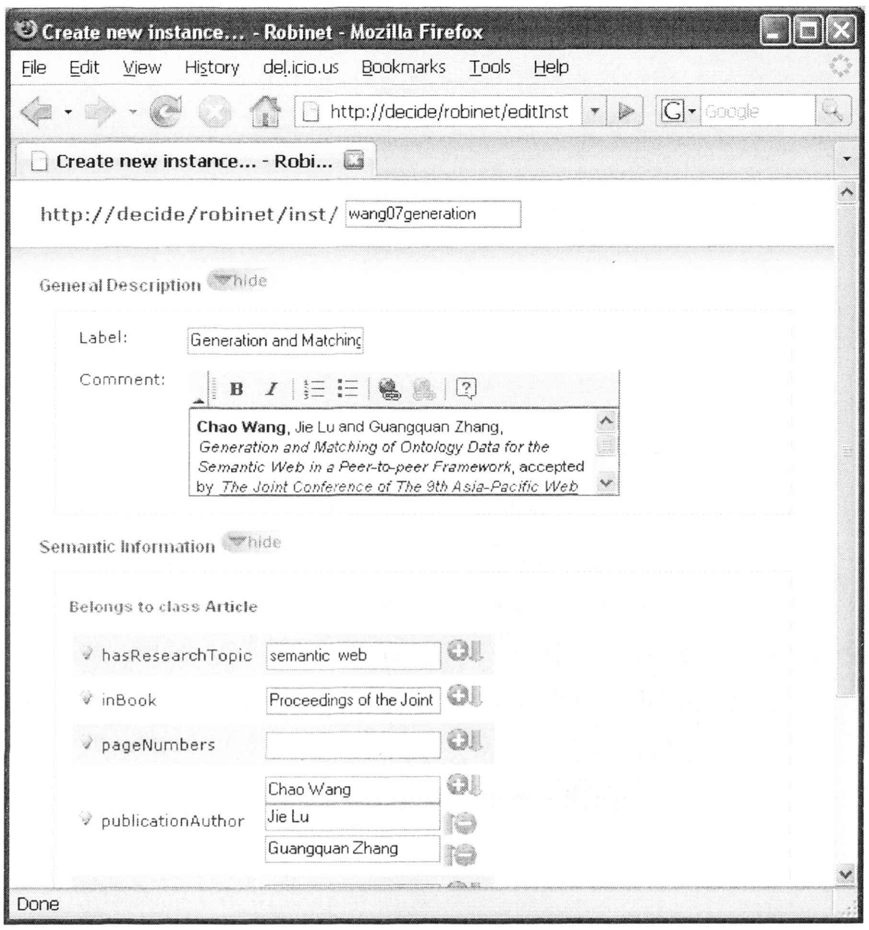
In addition, it is clear that the URLs used to identify their corresponding resources (classes or properties) in the system is quite friendly to both end users and other semantic web applications which may run at other computers in the network. For example, while the URLs `http://decide/robinet/onto/univ/` and `http://decide/robinet/onto/Article` lead to human-readable web pages about the ontology schema `univ` and the its class `Article` respectively, the URLs `http://decide/robinet/onto/univ` and `http://decide/robinet/onto#Article` (which can be accessed through their corresponding human-readable web pages as shown in Figure 4.4) identify the original OWL formatted version which is more machine-understandable. Therefore, other semantic web applications can use such URLs to import the interested ontology schemas into their application space remotely. This mechanism will help the growth of semantic web applications and intelligent software agents over the web.

4.3.3 Ontology Data Authoring

Once users has become familiar with the ontology schemas by browsing them, they can create new ontology data through the functions of the system. Currently, two main types of functions are implemented.

The *first* type of functions enables user to create completely new ontology data. For example, an academic staff, if he/she wants to publish some data about his/her recent publications, can choose the class that is most appropriate for the data. He/She may find the class `Article` in the ontology schema `univ` (as shown in Figure 4.4) is a proper class. Creating an instance of this class can be started by clicking the button "Create Instance" on that page. Figure 4.5 demonstrates the interface used for creating an instance of the class `Article`. An unique ID is required from the user to form the URL used to identify this publication. In addition, the user can supply some general information about the publication such as a human-readable label and some comments as well as the property values associated with the publication (e.g., `hasResearchTopic`, `inBook`, and etc). Some properties may have multiple values, for example, the author of the publication `publicationAuthor`. In such cases, the user can click the icon with the "plus" sign next to the concerned properties to add more values. By clicking the "Create" button at the bottom of the page, the data entered by the user will submitted to the system for the instance creation.

Figure 4.6 (a) shows the created instance in a human-readable view, which can be accessed through the URL generated based on the user-supplied ID. Actually, this view is generated by the system from the OWL formatted ontology data. Following the link of "http://decide/robinet/inst/wang07generation.owl", we can see the actual


Create new instance... - Robinet - Mozilla Firefox
File Edit View History del.icio.us Bookmarks Tools Help
Google
Create new instance... - Robi...
hide
hide
Done

http://decide/robinet/editInst

http://decide/robinet/inst/ wangO7generation

General Description

Label: Generation and Matching

Comment: **Chao Wang**, Jie Lu and Guangquan Zhang, *Generation and Matching of Ontology Data for the Semantic Web in a Peer-to-peer Framework*, accepted by *The Joint Conference of The 9th Asia-Pacific Web*

Semantic Information

Belongs to class Article

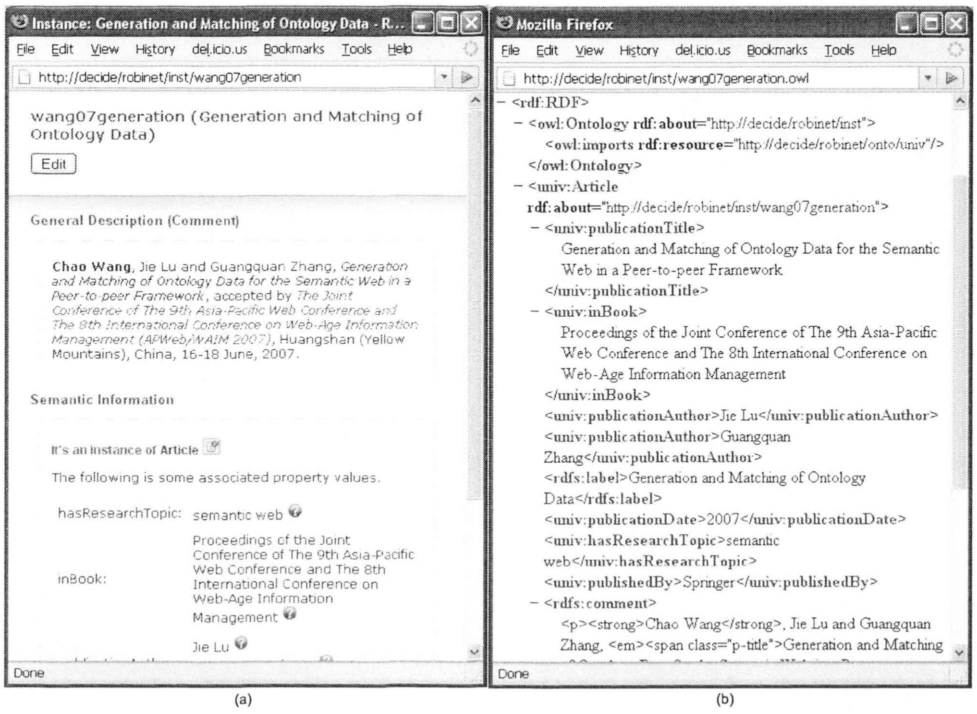
hasResearchTopic	semantic web
inBook	Proceedings of the Joint
pageNumbers	
publicationAuthor	Chao Wang
	Jie Lu
	Guangquan Zhang

Figure 4.5: The interface used for creating an instance of the class `Article`

ontology data as shown in Figure 4.6 (b). Obviously, the actual ontology data is not comfortable for ordinary users to read. However, computers on the other hand is good at processing such data. Since they are aligned with the given ontology schema, well designed semantic web applications that are aware of this domain ontology can use them to perform further complicated tasks.

The *second* type of functions allows users to create new ontology data which are related to the existing data.

First we discuss why we need this type of functions. This type of functions relies


Instance: Generation and Matching of Ontology Data - R...
File Edit View History del.icio.us Bookmarks Tools Help
Edit
Done
(a)
Mozilla Firefox
File Edit View History del.icio.us Bookmarks Tools Help
Done
(b)

http://decide/robinet/inst/wang07generation

wang07generation (Generation and Matching of Ontology Data)

General Description (Comment)

Chao Wang, Jie Lu and Guangquan Zhang, *Generation and Matching of Ontology Data for the Semantic Web in a Peer-to-peer Framework*, accepted by *The Joint Conference of The 9th Asia-Pacific Web Conference and The 8th International Conference on Web-Age Information Management (APWeb/WAIM 2007)*, Huangshan (Yellow Mountains), China, 16-18 June, 2007.

Semantic Information

It's an instance of Article

The following is some associated property values.

hasResearchTopic:	semantic web
inBook:	Proceedings of the Joint Conference of The 9th Asia-Pacific Web Conference and The 8th International Conference on Web-Age Information Management
	Jie Lu

http://decide/robinet/inst/wang07generation.owl

```
- <rdf:RDF>
  - <owl:Ontology rdf:about="http://decide/robinet/inst">
      <owl:imports rdf:resource="http://decide/robinet/onto/univ"/>
    </owl:Ontology>
  - <univ:Article
    rdf:about="http://decide/robinet/inst/wang07generation">
    - <univ:publicationTitle>
        Generation and Matching of Ontology Data for the Semantic
        Web in a Peer-to-peer Framework
      </univ:publicationTitle>
    - <univ:inBook>
        Proceedings of the Joint Conference of The 9th Asia-Pacific
        Web Conference and The 8th International Conference on
        Web-Age Information Management
      </univ:inBook>
      <univ:publicationAuthor>Jie Lu</univ:publicationAuthor>
      <univ:publicationAuthor>Guangquan
      Zhang</univ:publicationAuthor>
      <rdfs:label>Generation and Matching of Ontology
      Data</rdfs:label>
      <univ:publicationDate>2007</univ:publicationDate>
      <univ:hasResearchTopic>semantic
      web</univ:hasResearchTopic>
      <univ:publishedBy>Springer</univ:publishedBy>
    - <rdfs:comment>
        <p><strong>Chao Wang</strong>, Jie Lu and Guangquan
        Zhang, <em><span class="p-title">Generation and Matching
```

Figure 4.6: The created instance of the class `Article`. (a) The instance shown in a human readable view; (b) The actual ontology data created by the user.

on the existence of a type of properties called "object properties" which link one instance to another with a particular semantic relation (mentioned in Section 2.2.3). They offer the functionality as hyperlinks which are used over the current web to link different web pages together. The importance of hyperlinks has already been obvious. Without them, The web could not form and grow into such a huge connected information repository as currently is; nor could modern search engines build a large information base and rank the related information with certain measurements such as authority or popularity [94][23]. Similarly, it is also very important to use object properties to link instances together. The resulting relational feature in ontology data could be even more essential than using normal hyperlinks in web pages as it delivers more specific semantic meanings than normal hyperlink mechanism. Therefore, it is very desirable to enable users to create ontology data which are linked to other ontology data via certain object properties.

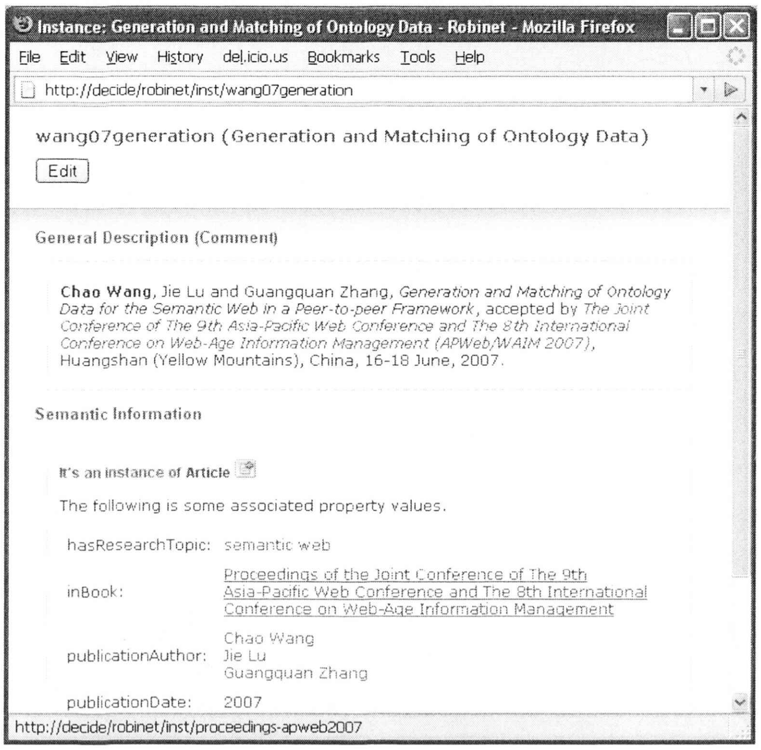
Then we show how easy it is to create related new ontology data based on existing ontology data in our system. As explained before, when a completely new instance is created, initially all the property values are supplied as strings by users, regardless of the types of properties. However, after the instance is created, the system will detect according to the ontology schema whether a property value should be a certain data type or an instance. For those who should be instances via given object properties, the system then provide the function to create them. As shown in Figure 4.6 (a), those property values that should be instances have question marks associated. By clicking the question mark, users will be presented an instance creation interface which is similar to the one shown in Figure 4.5. After creating these new instances, they are related to the existing instance with the specified object properties. Figure 4.7 shows the web pages of the existing instance after its related instances are created. The question marks has disappeared. Instead, those object property values are linked to the web pages of it's related instances.

4.4 Discussion

This section discusses the features of our web ontology data generation methods and their limitations.

The benefit of using XQuery for converting existing data is that large amount of ontology data can be created in this way. The limitation is that once the conversion has been made, any changes to the source data may not be reflected in the ontology data, therefore making the generated ontology data outdated. If changes to source data can not be avoided and happen very frequently, it is better to use a mapping mechanism instead of the conversion.

The web-based method for users to create web ontology data is quite user friendly.


Instance: Generation and Matching of Ontology Data - Robinet - Mozilla Firefox
File Edit View History del.icio.us Bookmarks Tools Help
http://decide/robinet/inst/wang07generation
Edit

wang07generation (Generation and Matching of Ontology Data)

General Description (Comment)

Chao Wang, Jie Lu and Guangquan Zhang, *Generation and Matching of Ontology Data for the Semantic Web in a Peer-to-peer Framework*, accepted by *The Joint Conference of The 9th Asia-Pacific Web Conference and The 8th International Conference on Web-Age Information Management (APWeb/WAIM 2007)*, Huangshan (Yellow Mountains), China, 16-18 June, 2007.

Semantic Information

It's an instance of **Article**

The following is some associated property values.

hasResearchTopic:	semantic web
inBook:	Proceedings of the Joint Conference of The 9th Asia-Pacific Web Conference and The 8th International Conference on Web-Age Information Management
publicationAuthor:	Chao Wang Jie Lu Guangquan Zhang
publicationDate:	2007

http://decide/robinet/inst/proceedings-apweb2007

Figure 4.7: The instance after related instances are created and automatically linked with it

It makes the complex and verbose format of web ontology data transparent to ordinary users. Furthermore, it also encourages the collaboration of creating ontology data from different users given the web-based platform. The limitations of this method may be its reliance on well-developed domain ontology schemas. The current implementation of the method doesn't allow the modification and extension of the ontology schemas that interpret the ontology data. Therefore, if the ontology schemas are not well developed when they are loaded into the system, the ontology data created by users later may not reflect the actual domains correctly.

4.5 Summary

This chapter presents two methods that deal with the generation of web ontology data. The first method uses XQuery to convert existing data which belong to a certain domain but have not yet been interpreted by certain ontology schemas to ontology data for given domains. The use of XQuery makes the conversion mechanism very flexible. The other method leads to a web-based prototype system that enables ordinary users to author new ontology data in a collaborative way similar to the popular (semantic) wiki systems. The overview of the system and its main functions have been discussed to show how easy users can use it to publish ontology data. Unlike other semantic wiki systems, this system can be extended with more management functions based the underlying ontologies, which will be discussed in the next chapter.

Chapter 5

Web Ontology Data Management

5.1 Introduction

In addition to the generation of web ontology data by authoring from ordinary users (as discussed in Chapter 4), proper management of such data is equally important, especially in the dynamic and open web settings. Many current wiki and semantic wiki systems encourage users to add new contents by using very few restrictions on users' operations. This open policy makes them very vulnerable to malicious actions from some ill-disposed users. It is therefore quite desirable to have a flexible management mechanism to protect such contents and data from unwanted actions, especially in some serious situations when data quality needs to be ensured to certain extent.

Some interesting situations may include the following (of course there exist more): (1) an IT company wants to have the information of its employees, divisions, projects, documents and etc managed collaboratively through a web based platform but hopes to maintain a certain level of security and privacy. (2) a department at a university hopes that its web site can be updated by its staffs, professors and students so that the web site reflects the latest information about professors (contacts, publications, courses to be offered, etc), students (contacts, courses enrolled in), and other resources.

The main goal of this chapter is then to propose an ontology enhanced model to enforce a flexible management mechanism on the system developed for users to author web ontology data. This mechanism should be suitable for the above situations where wikis and semantic wikis may not be competent. The resulted system should allow that ontology data (semantic contents) to be created and maintained collaboratively as in semantic wiki systems. But flexible restrictions should be imposed to ensure the contents are reliable and protected from malicious editing. In addition, since the contents to be managed are rich in semantics, the management should take advantage of this feature. That is, we try to use ontology techniques to make management itself rich in semantics, which differs from other conventional content management systems. We call this type of systems "semantic web content management systems" (SWCM systems).

To do so, we model a general SWCM system to be a composition of the following four key components: semantic contents, participants, operations and rules. From this model, semantic wiki systems are actually a special type of general SWCM systems with very weak rules. For the situations discussed above, a common type of SWCM systems is identified. In this type of systems, semantic contents are often made up of relatively stable domain ontology schemas and changing ontology data. The key components of this type of SWCM systems are discussed in detail to show how they can be modeled with semantic web techniques.

The rest of the chapter is organized as follows: Section 5.2 gives a definition of the model for general semantic web content management and discusses semantic wikis and a common type of SWCM systems according to this defined model briefly. Section 5.3 then discusses the common type of SWCM systems in more detail with respects to their four key components through a running example. In Section 5.4, we discuss design and implementation issues of our implemented system. A further discussion is given in Section 5.5 to show the advantages and limitations of this system. A possible road map is also suggested in this section. Section 5.6 summarize this chapter.

5.2 A General Semantic Web Content Management Model

We model a SWCM system at a certain level of abstraction. A general SWCM system is a 4-tuple $\Lambda(C, O, P, R)$ where C, O, P, and R are discussed in the following:

C (*Semantic contents*): the set of ontologies and related contents controlled by ontologies. As discussed before, ontologies can be made up of two parts: ontology schemas and ontology data. Besides the two parts as semantic contents, a SWCM system may have additional data such as images, pdf documents or other supporting files controlled or interpreted by ontologies.

O (*Operations*): the set of operations on semantic contents. The following forms the basic set of operations: {`create, edit, delete, view`}. For different applications or domains, concrete meanings will be associated to them. In addition, there can be more specific operations with typical semantics. For example, in a blog application, we may have a "comment" operation on the articles published by bloggers. In the domain of company information management, we may have a "change salary" operation on the instance of employees. Such operations can be viewed as special types of the four basic operations.

P (*Participants*): The set of different types of users who participate in the system by contributing and sharing contents through the set of operations O.

R (*Rules*): the set of rules that control the operations in O performed by participants on the contents. An example rule in R may look like this: "if x is an employee, then x can edit contents created by himself/herself". In the following sections, we will discuss how rules in R are defined and used for content management in more detail.

Based on this model, we can add more advanced functions such as query and searching. We will discuss this extension later in Section 5.5.2.

5.2.1 Semantic Wikis

Semantic wikis can be viewed as a special type of SWCM systems according to our model. The managed contents in these systems are mostly ontology schemas and data. Operations generally include "create", "edit", "delete" and "view" upon the contents. The rules to control the operations in these systems are simplified to a general policy that almost every one can edit every thing. By using certain authentication systems, new rules can be added into such semantic wiki systems. Some examples of rules may look like "only registered users are allowed to edit contents", "only administrators are allowed to delete contents". Such rules are not flexible for a complex domain that needs more fine grained restrictions with a certain sense of semantics.

5.2.2 Data focused Management

We discuss a common type of management of semantic web contents which focuses more on ontology data than on schema. In a general SWCM system, ontology schemas are also the targets for management. In systems like "OntoWiki" [10], participants can add, edit, delete classes and properties with very high freedom. This feature is good for ontology schema development. However, for a certain domain or application context with ontology schemas already developed, the requirement for frequent changes of them may not be very necessary. Particularly, since ontology schemas act as a set of common vocabulary for different parties, changing them frequently is not appropriate when they have been deployed (managing them through extension and evolution is more acceptable then). In contrast with ontology schema, ontology data is more variable. Management of ontology data therefore is more significant in such situations. For example, in the mentioned situation of company information management, the domain knowledge in term of concepts and relations within a company is relatively stable while the instance data often changes.

We call this common type of SWCM systems *data focused SWCM systems*. In the following sections, we will discuss them in detail. The situation of company information management will be served as a running example to make the discussion more clear.

5.3 The Data focused SWCM Model

5.3.1 Semantic Contents

In a data focused SWCM system, ontology data is the main target for semantic content management. This does not mean that ontology schema is not important. Obviously, a well modeled ontology schema that naturally reflects the domain knowledge casts rich semantics to ontology data, which increase the understandability of the system. The main task of ontology schema modeling, or usually referred to as ontology engineering [145], is to define concepts and relations that can reflect the target domain while retaining their relative independence of particular applications [144]. There exist several tools (e.g., Protégé [60]) that can help ontology engineers perform this task. Some approaches also exist, such as ontology learning [107]. They can help generate ontology (schema) at a certain level of automation. With these tools and approaches, people are able to develop a good enough domain ontology schema as the basis for a data focused SWCM system.

Given a set of basic domain ontology schemas, data focused SWCM system should be still able to manage the evolution and extension of them. Sometimes the basic domain ontology schemas fail to capture the domain knowledge with appropriate accuracy; or the domain may have some changes due to some certain aspects. In this case, participants can create new classes and properties as extensions to basic ontology schemas through the SWCM system. These new classes and properties can be mapped to the basic ontology schemas to achieve ontology evolution.

5.3.2 Participants

Participants in the system can also be modeled in ontology schemas in terms of types and relations. Particularly, for some domains, they are an essential part that is closely related to other aspects of the domains. For example, there can be a basic ontology schema $S_{company}$ describing structures of the company and resources in it. Participants of the SWCM system that uses this basic ontology schema in most cases are employees, whose types and relations can be modeled within $S_{company}$.

When participants of a SWCM system are modeled in ontology schemas (whenever in domain ontologies or in separate ontologies) and these ontology schemas form the basic schemas for this system, the instances of different types of participants actually become the contents that are manageable through the system.

To ease the further discussion in the following sections, we give some fragments of a sample ontology schema reflecting the domain of company information management as shown in Figure 5.1 (referred to as $S_{company}$ in following sections). They are defined with the notations from Description Logics. The symbol $\sqsubseteq$ in the figure denotes an is-a relationship between two concepts. For example, `Report` is a sub concept of `Document`. The symbol $\sqcap$ denotes a conjunction of concepts. For example, the fact "$\texttt{Employee} \sqsubseteq \texttt{Person} \sqcap \exists \texttt{workFor.Company}$" means that a developer should be a person who must work for at least one company.

We can see that the types and relations of participants of the SWCM system based on this ontology schema have been modeled into the ontology schema, from abstract concept `Employee` to more detailed concept `Developer`.

5.3.3 Operations

In a SWCM system, operations are also represented in ontologies.

First, we discuss the possibility of modeling operations in ontologies. In data focused SWCM systems, the targets of operations are mostly limited to instances of classes. For example, the target of the operation `create` is likely to be an instance of

$$
\begin{aligned}
\texttt{Company} &\sqsubseteq \texttt{Organization}\\
\texttt{Division} &\sqsubseteq \exists\texttt{partOf.Organization}\\
\texttt{Group} &\sqsubseteq \exists\texttt{partOf.Division} \sqcap \exists\texttt{member.Person}\\
\texttt{Employee} &\sqsubseteq \texttt{Person} \sqcap \exists\texttt{workFor.Company}\\
\texttt{Manager} &\sqsubseteq \texttt{Employee}\\
\texttt{ProjectManager} &\sqsubseteq \texttt{Manager} \sqcap \exists\texttt{manage.Project}\\
\texttt{FinancialStaff} &\sqsubseteq \texttt{Employee}\\
\texttt{Developer} &\sqsubseteq \texttt{Employee} \sqcap \exists\texttt{memberOf.Group}\\
\texttt{Report} &\sqsubseteq \texttt{Document}\\
\texttt{ProjectDocument} &\sqsubseteq \texttt{Document} \sqcap \exists\texttt{belongTo.Project}\\
\texttt{ProjectReport} &\sqsubseteq \texttt{Report}\sqcap\exists\texttt{belongTo.Project}\\
\texttt{Specification} &\sqsubseteq \texttt{ProjectDocument}
\end{aligned}
$$

Figure 5.1: Fragments of an ontology schema reflecting the domain of company information management

a certain class. On the other hand, the subjects of operations are obviously instances of certain types of participants, which are modeled with ontologies as well. Therefore, operations can be treated as object properties (as defined in OWL [111]) that link between contents and participants.

Then we discuss the types of operations. Although for different domains, there can be different operations with their own semantics specific to the domain, most of them can classified into the four basic operations we've mentioned before. For example, in the domain modeled with $S_{company}$, an operation that performs the registration of a new employee is actually a `create` operation upon the class `Employee`. An operation to change the salary of an employee performed by a financial staff is a fine grained `edit` operation. Correspondingly, such domain-specific operations can be modeled as sub-properties of the four basic operations. By analyzing the participants and targets of these operations, the relations between them can be determined. This results in a hierarchy of operations captured in the ontologies that is to be used in the SWCM systems.

5.3.4 Rules

Rules in a data focused SWCM system are used to control the operations on the contents. Unlike operations and participants, they are not modeled into ontology schemas in our method. This is partly because ontology languages, which are based on description logics, have several limitations [117]. Adding a rule layer on top of ontology is a common practice and efforts have been made to define rule languages for the semantic web such as SWRL [84]. By using rules, we are able to use logic programming (LP) combined with description logics [64] to perform reasoning tasks.

The following is some examples of rules to control operations in regard with $S_{company}$. These rules are expressed in an intuitive way for demonstration only. For example, Rule (5.3.1) states that any employee can edit contents of his or her instance[1]. Rule (5.3.3) states that a developer can edit a specification if he/she works on the project the specification belongs to. During implementation, these rules are recoded using prolog clauses, which will be discussed later.

[1]`edit`$(?x, ?x)$ may look somewhat confusing. We explain that the first $?x$ indicates the employee described by the instance while the second $?x$ represents the contents of the instance.

$$\begin{aligned}&\texttt{Employee}(?x)\\&\qquad\Rightarrow \texttt{edit}(?x, ?x)\end{aligned} \tag{5.3.1}$$

$$\begin{aligned}&\texttt{Developer}(?x) \wedge (\exists y)\texttt{workOn}(?x, y)\\&\qquad\Rightarrow \texttt{create}(?x, \texttt{Specification})\end{aligned} \tag{5.3.2}$$

$$\begin{aligned}&\texttt{Developer}(?x) \wedge \texttt{Specification}(?y)\\&\quad\wedge \texttt{belongTo}(?y, ?z) \wedge \texttt{workOn}(?x, ?z)\\&\qquad\Rightarrow \texttt{edit}(?x, ?y)\end{aligned} \tag{5.3.3}$$

There are two types of rules: permission and rejection (similar to the policy constructs "rights" and "prohibitions" in some policy languages [41, 91]). For example, Rule (5.3.4) is a rejection rule that doesn't allow the creation of any instances of `Employee` by developers.

$$\begin{aligned}&\texttt{Developer}(?x)\\&\qquad\Rightarrow \neg\texttt{create}(?x, \texttt{Employee})\end{aligned} \tag{5.3.4}$$

Rule Priority

If a large number of rules are defined to control operations, then conflictions between rules can hardly be avoided. Rule priority is used to partly resolve rule confliction. We assign a number between 0 and 10 as the priority to each rule. When two rules are found to conflict with each other, the rule with higher priority will override the other one. If conflicts still exist, human interference is necessary to resolve them before any permissions to be granted to the requested operations.

Reasoning Using Rules

We are able to define very flexible restrictions on operations thanks to the reasoning capabilities of ontologies together with rules. For example, given the ontology schema $S_{company}$, we can obtain Rule (5.3.5) indicating that a developer is an employee. Then we know that a developer can also edit his own instance using Rule (5.3.1).

$$\begin{aligned}&\texttt{Developer}(?x)\\&\quad\Rightarrow \texttt{Employee}(?x)\end{aligned} \tag{5.3.5}$$

As we've discussed, operations in a concrete domain can form a hierarchy. This implies that an operation that has a parent operation is more fine grained than its parent. Given implied rules like this, we are able to derive restricted operations on the targets. For example, the following rule set lets us know that an employee can edit his own instance but can not change his own salary given that Rule (5.3.7) has higher priority than Rule (5.3.6).

$$\begin{aligned}&\texttt{Employee}(?x)\\&\quad\Rightarrow \texttt{edit}(?x, ?x)\end{aligned} \tag{5.3.6}$$

$$\begin{aligned}&\texttt{Employee}(?x)\\&\quad\Rightarrow \neg\texttt{changeSalary}(?x, ?x)\end{aligned} \tag{5.3.7}$$

$$\begin{aligned}&\texttt{edit}(?x, ?x)\\&\quad\Rightarrow \texttt{changeSalary}(?x, ?x)\end{aligned} \tag{5.3.8}$$

$$\Rightarrow \neg\texttt{changeSalary}(?x, ?x) \wedge \texttt{changeSalary}(?x, ?x) \tag{5.3.9}$$

Rule Coverage and Engineering

Obviously, it is very difficult to create a complete rule set at once to cover all the operations properly according to a given security requirement if the domain is very complex. Sometimes, even the security requirement itself may not be very clear initially. In this case, we need to do rule engineering to approach the ultimate security goal. The process of rule engineering is also a way to check the validity of the rule set.

We can first define a basic rule set with relatively low priority and then add new rules with higher priority to override some old rules. A reorganization of these rules (both old and new) should also be allowed whenever it is necessary. Through this continuing process, we can finally meet the desired security requirement. That is, different participants are assigned with proper operation restrictions using rules so that unwanted accesses and operations are avoided.

According to different situations, we can have two approaches. (1) For a situation when security is considered very seriously, we start with a set of very rigid rules and then adjust them thoughtfully by adding new permission rules or relaxing exiting rejection rules. (2) In a very causal situation, we can start with rules that permit almost every operations. More rejection rules can be added in as the system evolves.

5.4 System Design and Implementation Issues

In Chapter 4, we have discussed the prototype system *robinet* for web ontology data generation. This prototype system has been extended to support our notions of data focused SWCM. This section overviews the design of the extended version of robinet and discusses some implementation details.

5.4.1 System Overview

Figure 5.2 illustrates the general structure of the extended *robinet* system and the interaction between the modules within it.

The part within the dotted lines in Figure 5.2 represents the components and modules used to implement the function of access or operation control. The general

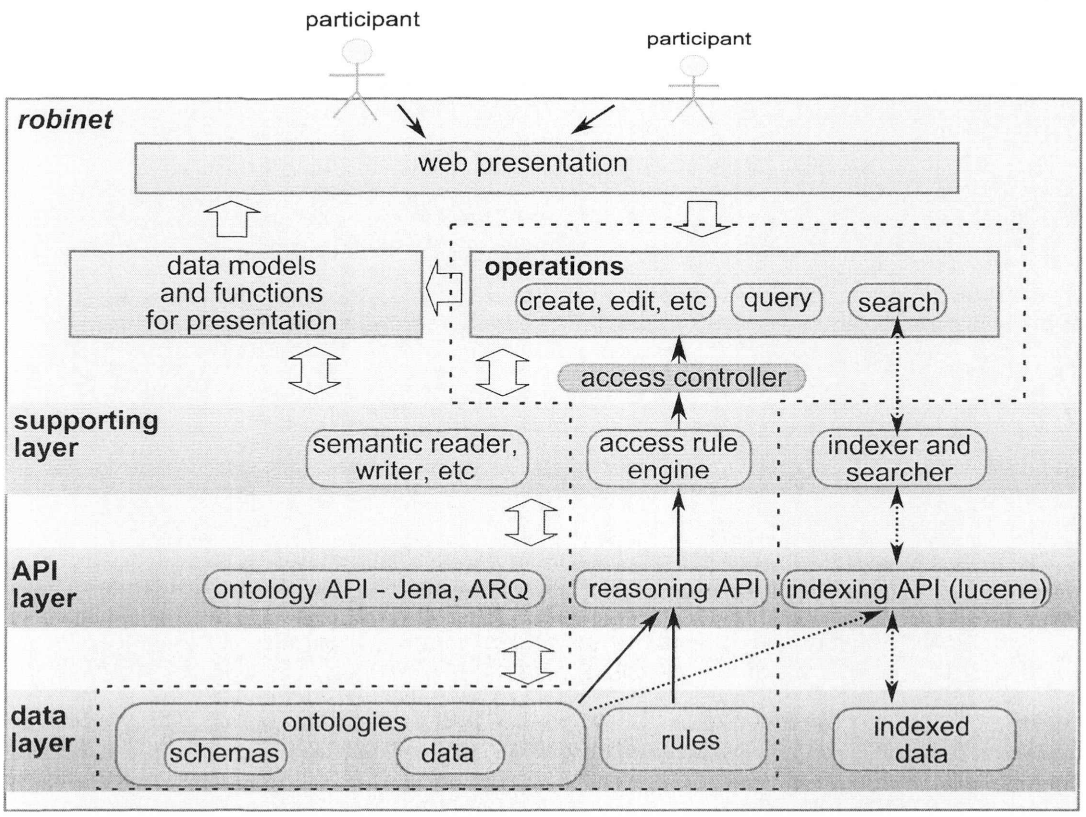


Figure 5.2: The general structure of the extended *robinet* system that supports SWCM

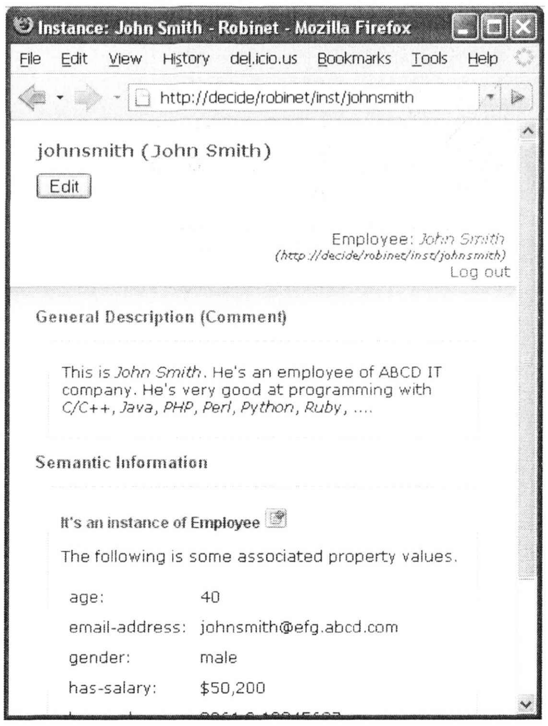
process for the operation control over the participants is as follows. First, when a participant tries to perform an operation over certain semantic content through the web interface, the information of the participant (i.e., its type), operation, and the content is retrieved by consulting the working ontologies. This information, together with the rules, is processed through the reasoning APIs by the rule engine. Currently, we use prolog to perform the reasoning task. Through the reasoning, whether the operation is accepted or rejected will be known [2]. When a conflict happens, the rule priorities will be used for further decision. When the conflict still remains, an error message will be logged for further review. Depending on the final result generated

[2]Some default rules will be specified to avoid non-coverage.

by the rule engine, the access controller determines whether or not to proceed the requested operation.

To bootstrap its semantic content management, the system initially needs a set of adequate domain ontology schemas, instance data and rules. These should be provided by the administrator of the system in advance. Sometimes, a few domain specific operations should be implemented and configured as well.

Inherited from the original *robinet* system, it uses friendly URLs as well. In particular, each instance is identifiable through a URL, as shown in Figure 5.3. The URL in this figure links to the page about an employee. And it is also used to identify this employee. The use of URL as instance identity allows us to handle them with ease.


Instance: John Smith - Robinet - Mozilla Firefox
File Edit View History del.icio.us Bookmarks Tools Help
http://decide/robinet/inst/johnsmith
Edit
Log out

johnsmith (John Smith)

Employee: *John Smith*
(http://decide/robinet/inst/johnsmith)

General Description (Comment)

This is *John Smith*. He's an employee of ABCD IT company. He's very good at programming with *C/C++*, *Java*, *PHP*, *Perl*, *Python*, *Ruby*, ...

Semantic Information

It's an instance of Employee

The following is some associated property values.

age:	40
email-address:	johnsmith@efg.abcd.com
gender:	male
has-salary:	$50,200

Figure 5.3: A web page that shows the content of an instance of the class `Employee`. The URL is also used to identify this employee as he's logged in.

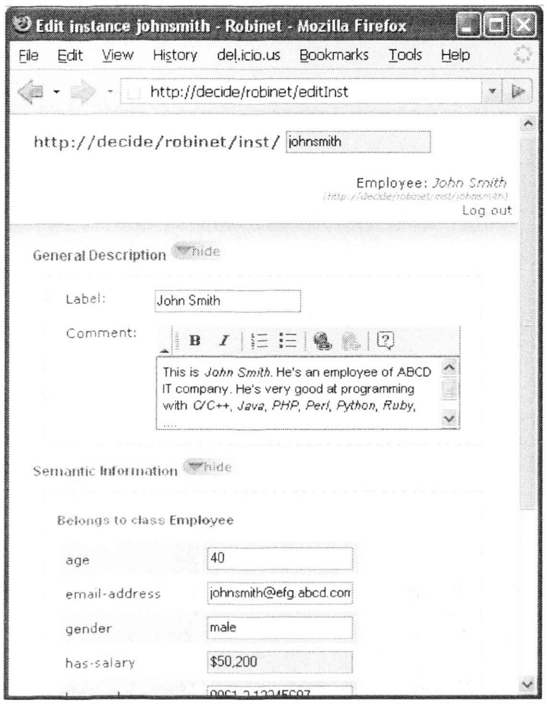
5.4.2 On Implementing Operations

Operations in the system are also identified by URLs. For example, "http://decide/robinet/editInst" represents the `edit` operation; while "http://decide/robinet/change Salary" represents the `changeSalary` operation. In this case, several issues should be addressed.

First, there exists an overlap between the friendly URLs for instances and the URL for `view` operation as each instance URL actually implies a `view` operation on it. Our system maps all the requests to instance URLs to the `view` operation. This mapping prevents unrestricted access to the instance contents which may be protected by rules on `view` operation.

Second, some domain specific operations need to be configured to work properly. For example, when we specify that `changeSalary` is a sub operation of the `edit` operation on instances of `Employee`, we need to associate the set of properties that can be edited by `changeSalary` with it. This enables the system to know which can be changed when `changeSalary` is permitted by reasoning using the rules (or which can not be changed if it is rejected). Generally, it allows the system to calculate the sub set of properties it can operate on through the rules and the additional associated information. For example, when the employee requests to edit his own instance data, he's not allowed to change his own salary, as shown in Figure 5.4, according to the rule.

Finally, some domain specific operations may still have to be implemented through simple programming. For example, a company may need a `promote` operation which means to increase a certain amount of salary for an employee due to the salary system of this company. This typical operation then needs to be programmed and configured into our system.


Edit instance johnsmith - Robinet - Mozilla Firefox
File Edit View History del.icio.us Bookmarks Tools Help
hide
hide

http://decide/robinet/editInst

http://decide/robinet/inst/ johnsmith

Employee: *John Smith*

Log out

General Description

Label:	John Smith
Comment:	This is *John Smith*. He's an employee of ABCD IT company. He's very good at programming with *C/C++*, *Java*, *PHP*, *Perl*, *Python*, *Ruby*, ...

Semantic Information

Belongs to class Employee

age	40
email-address	johnsmith@efg.abcd.com
gender	male
has-salary	$50,200

Figure 5.4: A web page showing that the value of `has-salary` can not be changed by the employee himself according to the imposed rules

5.4.3 On Implementing the Rule Engine

We use prolog to implement the rule engine. The knowledge reflected by the domain ontology schema is asserted into a prolog database for reasoning. For example, the is-a relation between the concepts `Developer` and `Employee` is asserted as a clause "c_Developer(X):-c_Employee(X).". If a new instance of `Developer` "John" is created, it will be asserted as a fact like "c_Developer(i_John).".

As mentioned before, rules are recoded into prolog clauses and their priorities are considered as well. The head of the rule clause is a 5-arity functor "accept" or "reject", representing the two types of rules. For example, the rule clause "accept(P, edit, C, 2, all):-c_Developer(P), c_Specification(C), p_belongTo(C, T1), p_workOn(P, T1)." means that a developer can edit all the properties (what "all" in the clause indicates) of a specification when he/she works on the project the specification belongs to. This actually implements Rule (5.3.3). This rule has an priority of two and can be overridden by rules with priorities higher than two.

A prolog interpreter "JLog" [3] is used to interpret and infer on the database of facts and clauses. After the rule set is tested and validated through the rule engineering, access permissions can be consulted through the prolog interpreter.

5.5 Further Discussion

This section provides some discussions on comparison of this system with other types of systems. It also discusses how other desirable functions can be added to the system.

5.5.1 Comparison with Other Systems

We discuss advantages and limitations of the proposed data focused SWCM system in comparison with semantic wiki systems and conventional web based content management systems.

The advantages over semantic wiki system is obvious. As we've discussed before, reliability and security is a major problem of semantic wiki systems. Our system uses rules to semantically restrict operations on contents according to different participants. This increases reliability. Although collaboration may be restricted by this mechanism, this restriction actually is necessary in certain domains. By this restriction, we expect participants to produce high quality semantic contents. In addition, by editing rules, we can achieve different security levels with great flexibility.

Conventional web and database techniques can be used to implement a similar system to perform content management, which is actually a common solution nowadays. Compared to this solution, to develop a content management system based on our system for a certain domain, the major tasks is: (1) to model the domain ontology schemas; and (2) to define rules restricting operations on data. Once these tasks

[3]http://jlogic.sourceforge.net/

are performed successfully, the system is almost ready. Less programming tasks may be involved. In addition, while conventional content management systems require additional approaches to turn their managed contents into formats conforming to semantic web standards, our data focused SWCM system manages semantic contents directly. Therefore, it is more compatible with the emerging semantic web.

There may exist some limitations of our system as well. To let our system work properly, basic ontology schemas for the system need to be well developed so that they can reflect the domains appropriately. However, ontology development and engineering are relatively new to most of ordinary developers. In addition, while the rule system can give us a lot of flexibility, it also implies that efforts need to be made to create a proper set of rules for the required security requirement. Hopefully, we expect that the spread of semantic web technology and the further research in this area will change the current situation.

5.5.2 Towards Semantic Web Knowledge Management

This subsection discusses how query, search and other advanced functions can be added upon the data focused SWCM systems. In particular, we suggest a layered approach as depicted in Figure 5.5. In this approach, basic semantic content management functions such as collaboration, access control, and etc are first implemented upon the semantic web content to be managed. Built upon these basic management functions, some simple query and search functions may be provided to meet the information needs of human participants. Furthermore, topics such as advanced query and search, reasoning, and etc may be studied as the topics of knowledge management. They are expected to offer a higher level interface to human participants and software agents.

We haven't discussed query and search in our basic data focused SWCM system. Actually they are very essential. As shown in Figure 5.2, simple search function is designed and implemented using the tool Lucene [4]. This function allows users to

[4]Lucene, http://lucene.apache.org

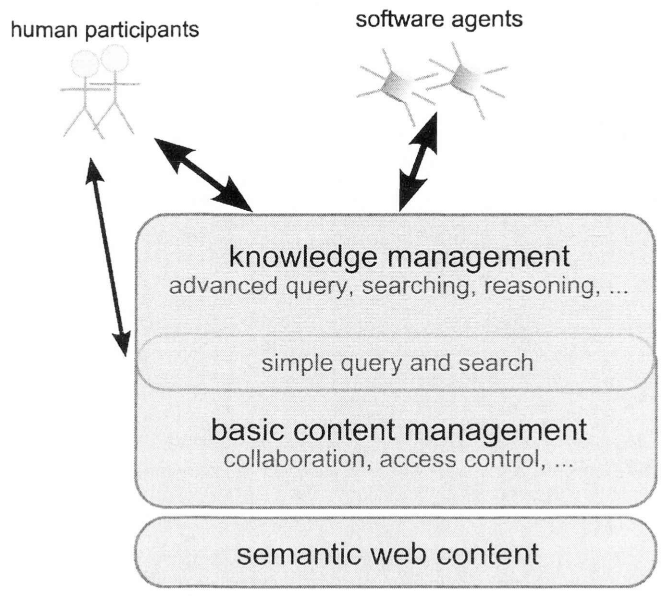


Figure 5.5: The semantic web knowledge management over content management

perform full text search on the instance data. Tools that support query languages such as SPARQL [130] are also employed to implement simple structured query functions. It is then possible to develop more advanced search and query functions. Such advanced functions, together with reasoning and other functions, form the research topics of building semantic web knowledge management systems.

5.6 Summary

In this chapter, we propose a model for general semantic web content management (SWCM) systems and view the recent semantic wiki systems as a special type which can not be used in some serious situations due to its limitation. Therefore, data focused SWCM systems that can be used for such situations are proposed. Their key components are discussed to show how they can be modeled. We discuss our implemented system to show its design considerations, general structures and certain implementation issues. Its advantages over semantic wiki systems and other conventional web based content management system are discussed in terms of reliability, security, flexibility and compatibility with the emerging semantic web.

Chapter 6

Ontology Data Matching by Constrained Clustering

6.1 Introduction

From this chapter on, we will discuss the third issue related to ontology-based data integration: "How to improve the quality of integrated ontology data by reducing duplications and increasing completeness and certainty."

As discussed in the previous chapters, more and more convenient methods and tools are becoming available for people to develop ontology schemas and publish ontology data. This trend could eventually lead us to a distributed web platform where data published from different individuals or organizations are more semantically understandable (for computers) due to the conformance to certain domain ontologies. While a particular domain ontology acting as a global schema makes data integration much easier, the duplication problem caused by overlapping data may become even more serious because of the increasing amount of sources and more decentralized environment. Therefore, the task of identification of duplicated ontology data within a certain domain becomes very necessary if we want to improve the quality of data and obtain reliable analytical results upon it or create reliable services from it. The detailed scenario will be discussed to show this necessity in this chapter.

The issue of duplicated data is not an uncommon problem. It happens in traditional data integration as well as in ontology data integration. Since this problem degrades the quality of the integrated data, affecting the tasks performed on the data such as data mining and knowledge discovery, duplicate detection/elimination has then become an essential preprocess. Different methods have been proposed to deal with this problem [56, 172, 33, 18]. The main idea of these methods is to use certain metrics to determine if certain pairs of data records are similar enough to be duplicates. These methods are mostly designed to tackle the duplication problem in the traditional data integration setting, in which each data record is mostly of one same type and exists as an independent instance during the duplicate detecting process. In contrast, ontology data have more features, which we believe can help develop even more effective methods to tackle the duplication problem in them.

This chapter then proposes an efficient clustering-based approach to detect duplicates among ontology data. It takes advantage of one typical feature of ontology data: *relational feature*. In addition, the characteristics of ontology data are analyzed from the perspective of duplicate detection. We define constraint rules that capture these characteristics. Our approach then incorporates these constraint rules into a typical canopy clustering process for duplicate detection. Experiments show that our approach performs well with improved accuracy. Furthermore, as our approach is based on clustering, no labeled samples are essentially required and no extra training process is involved, which sometimes is good for large and raw data sets.

The rest of the chapter is organized as follows. Section 6.2 shows several scenarios when ontology data matching is desirable. Characteristics of ontology data are discussed in Section 6.3. Section 6.4 defines a set of constraint rules for duplicate detection in ontology data. Section 6.5 presents a constrained clustering approach. Related experiments and evaluation results are shown in Section 6.6. Section 6.7 summarizes this chapter.

6.2 Scenarios

We will first discuss a detailed scenario related to the semantic web. Then some other possible scenarios will be shown.

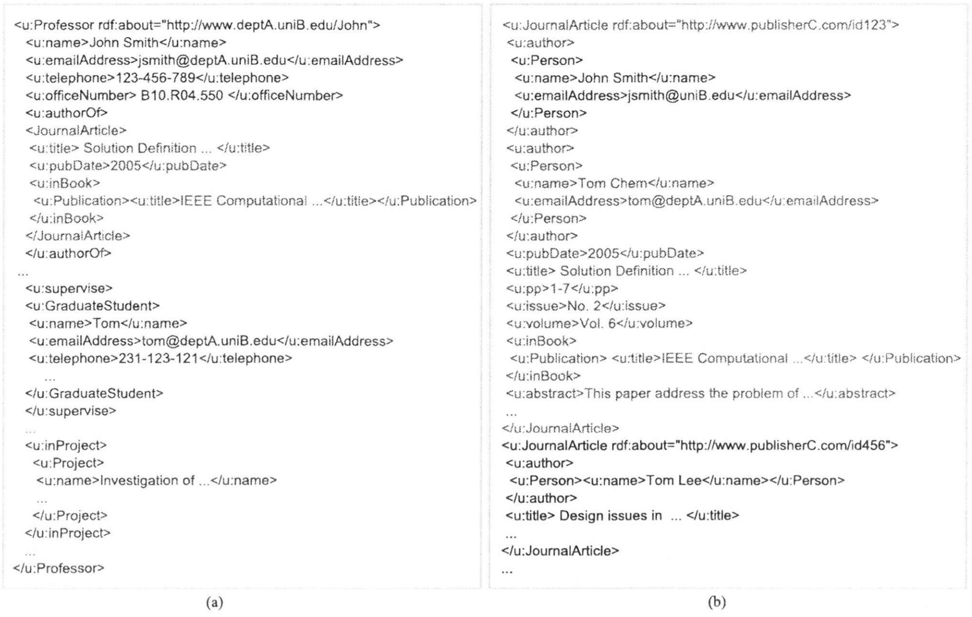
6.2.1 A Scenario Related to the Semantic Web

Since the semantic web is a distributed environment, we picture it as an environment consisting of numerous information sources or peers. For a particular domain, there could be one or more ontologies that encode the knowledge of this domain in the form of concepts, properties and their semantic relations. Let's assume only one such backbone ontology for one domain in this study. This ontology is actually recognized as Terminology Box (TBox) or ontology schema as we've discussed. For such a domain ontology, a single data source can hardly offer entire or enough data in the form of individuals or instances for the concepts of the ontology. In reality, each data source provides a portion of data that is aligned to a certain facet of the ontology. This facet reflects the particular view and information usage of the peer that creates and maintains the data source.

The detailed scenario could be as follows. A professor contributes an information source about his/her own details including contact information, supervised students, research project, publications, and etc. Meanwhile a publisher contributes another information source consisting of a detailed publication list, which includes some publications of that professor. Figure 6.1 illustrates an example of possible ontology data contributed by the two sources. As assumed, a common ontology schema is used in the scenario.

It is easy to see that the publisher's source lets us know details of publications by that professor. But it can not offer us information about the research project of the professor, neither the information about his supervised students, while the professor's information source can offer.

In the current web setting, it is still possible to retrieve both of the sources about



```
<u:Professor rdf:about="http://www.deptA.uniB.edu/John">
  <u:name>John Smith</u:name>
  <u:emailAddress>jsmith@deptA.uniB.edu</u:emailAddress>
  <u:telephone>123-456-789</u:telephone>
  <u:officeNumber> B10.R04.550 </u:officeNumber>
  <u:authorOf>
  <JournalArticle>
   <u:title> Solution Definition ... </u:title>
   <u:pubDate>2005</u:pubDate>
   <u:inBook>
    <u:Publication><u:title>IEEE Computational ...</u:title></u:Publication>
   </u:inBook>
  </JournalArticle>
  </u:authorOf>
 ...
  <u:supervise>
  <u:GraduateStudent>
   <u:name>Tom</u:name>
   <u:emailAddress>tom@deptA.uniB.edu</u:emailAddress>
   <u:telephone>231-123-121</u:telephone>
      ...
  </u:GraduateStudent>
  </u:supervise>
  ...
  <u:inProject>
   <u:Project>
    <u:name>Investigation of ...</u:name>
    ...
   </u:Project>
  </u:inProject>
  ...
</u:Professor>
```

(a)

```
<u:JournalArticle rdf:about="http://www.publisherC.com/id123">
 <u:author>
  <u:Person>
   <u:name>John Smith</u:name>
   <u:emailAddress>jsmith@uniB.edu</u:emailAddress>
  </u:Person>
 </u:author>
 <u:author>
  <u:Person>
   <u:name>Tom Chem</u:name>
   <u:emailAddress>tom@deptA.uniB.edu</u:emailAddress>
  </u:Person>
 </u:author>
 <u:pubDate>2005</u:pubDate>
 <u:title> Solution Definition ... </u:title>
 <u:pp>1-7</u:pp>
 <u:issue>No. 2</u:issue>
 <u:volume>Vol. 6</u:volume>
 <u:inBook>
  <u:Publication> <u:title>IEEE Computational ...</u:title> </u:Publication>
 </u:inBook>
 <u:abstract>This paper address the problem of ...</u:abstract>
 ...
</u:JournalArticle>
<u:JournalArticle rdf:about="http://www.publisherC.com/id456">
 <u:author>
  <u:Person><u:name>Tom Lee</u:name></u:Person>
 </u:author>
 <u:title> Design issues in  ... </u:title>
 ...
</u:JournalArticle>
...
```

(b)

Figure 6.1: An example of the ontology data in the semantic web scenario. (a) the instance about the professor from this professor's source; (b) the instances about publications from the publisher's source.

the professor through a web search engine. But some query string tuning techniques may be required and several queries are needed. Besides that, as the two sources come from different queries, or luckily from one query but surely different result items, users have to manually browse them separately to find out the desired information. We could manually create a hyperlink at a proper place to relate those two information sources so that people can follow the link to get more information when they browse. But under the vision of the semantic web, we want this happens automatically. That is to relate the instance of professor from the publisher's source to the professor's own instance automatically. Obviously, by correctly matching these two instances from different sources provided by the data-level integration function, better information services could be offered. For example, when a software agent reaches one of the information sources, more information can be requested through matched instances before it satisfies its special information needs.

6.2.2 More Scenarios in Other Areas

Besides the common scenario for ontology data matching in the semantic web setting, there could also be more scenarios in other significant areas such as security informatics, life science and health care.

Appropriate data integration and information sharing between different government agencies could lead to a more confident combat against the terrorism and crimes. The tragic event of September 11 has shown that the lack of effective information sharing and analysis was one of the major factors that prevent the USA government from avoiding this tragedy [126]. This problem may reflect several faults from different perspectives (e.g., organizational, managerial, technologic). From the perspective of technology, we suggest that the use of ontology related technology and the application of matching/linking methods could help solve the problem. The ontology related technology can render more semantics to the information produced by different government agencies so that these pieces of information can be integrated together with much ease. Then, matching can be performed on the integrated data to analyze if links can be found between two seemingly irrelevant instances, similar to the process shown in the scenario in the semantic web setting.

Scenarios could also exist in the area of health care. This area has already witnessed an increasing awareness of the ontology related technologies. Some ontology-like standards, for example, Health Level 7 (HL7) [1], have been developed. More and more health information is being recorded as some standardized forms of electronic medical records (EMR) [34]. Therefore, it is possible to integrate such EMRs from different sources. The matching process can then help identify the information among those EMRs that refers to the same patient, which could enable authorized doctors to make informed decisions.

[1] http://www.hl7.org

6.3 Characteristics of Ontology Data

One main characteristic of ontology data is the relational feature, i.e., the semantic links between different data instances. This often implies that instances have different types, like the discussed scenario where the author instance link to publication instances. Then, multi-type is another characteristic.

In the discussed semantic web scenario, data instances are not formatted as well as in databases. They are often presented in XML format or described by certain languages (for example, OWL [111]). Therefore, such data instances are semi-structured. In addition, as users can choose different ways to express, the resulting data instances then have different perspectives, not as unified as those in databases.

6.4 Duplication in Ontology Data

Duplication in ontology data can happen on every type of related data instances. However, due to the characteristics of ontology data, there are some certain patterns among them, which allow us to define constraints. We first introduce some basic notations and then define five constraint rules.

6.4.1 Notations

First, for a particular domain of interest, we can obtain a set of types T, and a set of properties P. There are two types of properties in P: data type properties that allow instances to be described with numbers and/or string values; and object properties that link instances to other instances with particular meanings (following the notions in OWL [111]). An instance then can be described with a type and a subset of properties and their corresponding values (numbers, strings, or other instances).

We identify two classes of instances. If an instance d_i has certain object property values that let it link to a set D_i of other instances, then d_i is identified as "*primary instance*". For any instance d_j ($d_j \in D_i$), d_j is identified as "*derived instance*".

The two classes are not exclusive. That is, an instance can be both "primary" and "derived" as long as it points to other instances and has other instance pointing to itself. Given an object property link between two instances (denoted by $d_i \rightarrow d_j$), it is easy to determine the classes of the instances.

If two instances d_i and d_j actually refer to one same real world entity, then the two instances are regarded as duplicates (denoted as $d_i = d_j$). Duplicated instances may not be same in terms of their types, property values as they often come from different sources with different qualities and perspectives. But usually they have similar values. Traditional methods thus use certain similarity measures to compute degrees of similarity of two instances. Given a similarity function f, a clustering process can be conducted to group instances with high similarity degrees into same clusters. For an instance d_i grouped into cluster c_k, we denote as $d_i \in c_k$ or simply $c_k(d_i)$.

6.4.2 Constraint Rules

We here define five constraint rules for duplicate detection using clustering approaches. Please note although we call all of them constraints, some actually act more like general rules with little constraint features.

Rule 1 (Derived distinction). Given an instance d_p and $D_p = \{d_r | d_p \rightarrow d_r\}$, if $\forall d_i, d_j \in D_p, i \neq j$, then $d_i \neq d_j$.

This rule indicates that all the derived instances from *one* same primary instance should not be duplicates of each other. The reason is quite obvious. Firstly, the application of relating one instance to two or more same other instances is very rare. A paper is always written by different authors if it has more than one author. A conference, in principle, never allows two same papers to be accepted and published. Secondly, the relation between one instance and other several instances often occurs within one data source. Therefore, it is quite easy to maintain so that the derived instances from one same instance are not duplicates. Consider that academic staffs manage their publications to ensure no duplicates occur on their web pages. As a result, the instances of the staffs links to different instances of publications.

Rule 2 (Primary similarity). Given two *primary* instances d_a, d_b and one of the resulting clusters c, if $c(d_a, d_b)$, then d_a and d_b have high confidence to be duplicates. We denote $d_a \approx d_b$.

This rule prefers similar primary instances. This rule is based on the observation of the characteristic that primary instances are often described with more detailed and accurate information while derived instances are usually given less attention and hence have less and vaguer details. Therefore, similarity between primary instances are more reliable for duplicate detection.

Rule 3 (Derived similarity). Given two *primary* instances d_a, d_b and $d_a \approx d_b$, if we have instances d_x, d_y and cluster c such that $d_a \longrightarrow d_x$, $d_b \longrightarrow d_y$, $c(d_x, d_y)$, then $d_x \approx d_y$.

This rule treats derived instances that fall in same cluster as duplicates if their corresponding primary instances are treated as duplicates. Strictly speaking, if two primary instances are duplicates, all of their corresponding derived instances should be duplicates as well. However, as noise often exists, it can not be guaranteed that the seeming primary instance duplicates are actual duplicates. To ensure high precision and to prevent false duplicate spreading, we only identify those derived instances that fall in same clusters to be duplicates.

Rule 4 (Reinforced similarity). Given instances d_i, d_j, d_m, d_n and clusters c_k, c_l, if we have $d_i \longrightarrow d_m$, $d_n \longrightarrow d_j$, $c_k(d_i, d_j)$ and $c_l(d_m, d_n)$, then $d_i \approx d_j$ and $d_m \approx d_n$.

This rule addresses the issue of data expressed with different perspectives. Different sources have their own views and describe data from different angles. An entity may be described as a detailed primary instance in one source; But in another source, it could be a simple derived instance. While we may not be confident in the similarity between a primary instance and a derived instance that fall in one same cluster c_k, this similarity will be reinforced if their derived/primary instances also fall into one same cluster c_l. As a result, we treat both pairs as duplicates.

Rule 5 (Boosted similarity). Given *derived* instances d_i, d_j, d_m, d_n and clusters

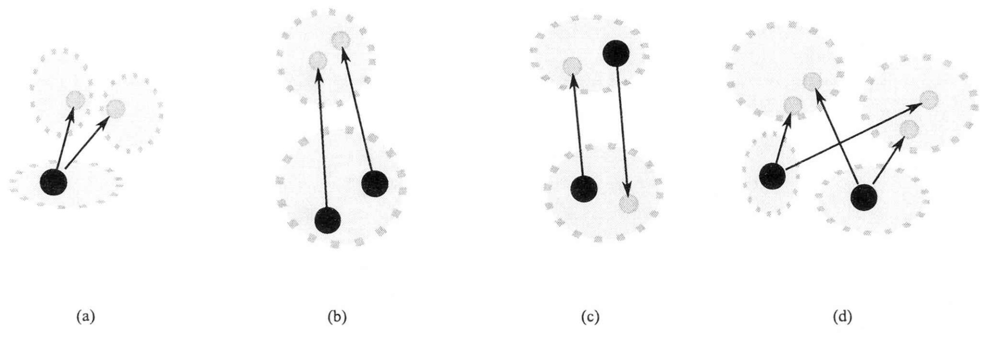


Figure 6.2: An illustration of applications of different constraint rules in corresponding situations. (a) derived distinction; (b) primary similarity and derived similarity; (c) reinforced similarity; (d) boosted similarity.

c_k, c_l such that $c_k(d_i, d_j)$ and $c_l(d_m, d_n)$, if there exist instances d_x, d_y such that $d_x \not\approx d_y$, $d_x \rightarrow [d_i, d_m]$ and $d_y \rightarrow [d_j, d_n]$, then $d_i \approx d_j$ and $d_m \approx d_n$.

This rule reflects the notion of co-referencing. It is possible that two different instances mention two seemingly same instances that turn out to be different. But the possibility would be much less if more than one (unique) instances mention two sets of seemingly same but different instances. For example, two different papers may have one author's name in common which actually refers to two different persons; But it rarely happens that two papers have two authors' names in common which refers to four different persons. Ideally, if more frequent primary instances are found pointing to more sets of similar derived instances (which may be implemented by frequent item set mining [2]), the confidence of the results would be much higher.

According to what we've discussed, Figure 6.2 illustrates the application patterns of different constraint rules that have been defined.

6.5 A Constrained Clustering Approach

This section discusses how the above rules are incorporated in the clustering process. First we present the commonly used canopy clustering method in duplicate detection. Then we focus on our proposed approach.

6.5.1 Canopy Clustering

Canopy clustering [109] is commonly used in duplicate detection [33, 18, 140]. It uses two similarity thresholds (T_{tight}, T_{loose}) to judge if an instance is closely/loosely similar to a randomly selected instance that acts as canopy center. All loosely similar instances will fall into this canopy cluster. But those closely similar instances will be removed from the list and never compared to another canopy center. Canopy clustering is very effective in duplicate detection as most instances are clearly non-duplicates and thus fall in different canopies. It is also very efficient since it often uses quick similarity measures such as TFIDF [137] computed using inverted index techniques.

Since the resulting canopies may be still large and overlap with each other, a second stage process such as Greedy Agglomerative Clustering (GAC) or Expectation-Maximization (EM) cluster are usually conducted within each canopy to yield refined results [109].

When canopy clustering is applied to duplicate detection in relational data, such as ontology data, directly, the performance may not be as good as it is used in normal data. This is because it ignores particular characteristics of these data. For example, for two derived instances which may represent two different publications of one person, they can be so similar that canopy clustering (even with GAC or EM) treats them as duplicates.

6.5.2 Canopy Clustering with Constraints

To improve the performance of duplicate detection in ontology data, we modified canopy clustering by incorporating the constraints we've defined. The resulting approach can be divided into four steps.

The first step (Step I) is much like the first stage of canopy clustering except that it subjects to the constraint that no any two derived instances from one same instance fall into one same canopy. Figure 6.3 shows the algorithm of this step. In

```
Input: Set of instances $D = \{d_1, d_2, \cdots, d_N\}$;
        Similarity threshold $T_{tight}, T_{loose}$.
Output: Set of canopy clusters $C_1 = \{c_1, c_2, \cdots, c_K\}$.
Begin
  $C_1 = \emptyset$; $D_{tmp} = D$;
  while $D_{tmp} \neq \emptyset$ do
    Create a new canopy cluster $c_{canopy} = \emptyset$;
    Pick a random $d_r$ in $D_{tmp}$;
    Let $c_{canopy} = \{d_i | d_i \in D_{tmp} \wedge sim(d_i, d_r) > T_{loose}\}$
      subject to condition:
        $\forall d_x, d_y \in c_{canopy}\ (x \neq y) \Rightarrow \{d_z | d_z \rightarrow d_x \wedge d_z \rightarrow d_y\} = \emptyset$;
    Let $c_{core} = \{d_i | d_i \in c_{canopy} \wedge sim(d_i, d_r) > T_{tight}\}$;
    $D_{tmp} = D_{tmp} - c_{core}$; $C_1 = C_1 + c_{canopy}$;
  End while
  Output $C_1$;
End
```

Figure 6.3: The algorithm of Step I

the algorithm, function $sim(d_i, d_r)$ computes the degree of similarity between the instance d_i and d_r.

Although each resulting cluster is constrained to contain no two derived instances of one same instance, it still can not guarantee **derived distinction** due to the existence of overlapping canopies. If two clusters, each of which contains a derived instance of one same instance, both have an instance $d_{overlap}$, this instance then actually bridges the two different derived instances when we take a transitive closure. As a result, it violates **derived distinction**.

Step II then is designed to ensure **derived distinction** thoroughly. It is done by checking the overlapping instances and only allowing them to be with the most similar derived instance. Figure 6.4 shows the algorithm of Step II.

The purpose of Step III is to extract high confident duplicate pairs within each cluster in C_2 by following the definition of **primary similarity**, **derived similarity**, and **reinforced similarity**. In Step IV, **boosted similarity** is implemented

```
Input: Set of instances $D = \{d_1, d_2, \cdots, d_N\}$;
       Set of clusters $C_1$ generated from Step I.
Output: Set of clusters $C_2$.
Begin
  for each $d_i \in D$ do
    $D_{derived} = \{d_j | d_i \rightarrow d_j\}$;
    for any $d_x, d_y \in D_{derived}$ $(x \neq y)$ do
      if $\exists d_z \in D,\ c_m, c_n \in C_1$ such that $d_z, d_x \in c_m$ and $d_z, d_y \in c_n$
        let $\delta = sim(d_z, d_x) - sim(d_z, d_y)$;
        if $\delta > 0$ then remove $d_z$ from $c_n$ else remove $d_z$ from $c_m$;
      end if
    end for
  end for
  Output $C_1$ as $C_2$;
End
```

Figure 6.4: The algorithm of Step II

to extract frequent co-referenced instance pairs as potential duplicates from the clusters. The algorithms of Steps III and IV are illustrated in Figure 6.5 and Figure 6.6 respectively. After all the potential duplicate pairs are extracted, a transitive closure is performed to generate the final results.

Please note the constraint rules reflected in these steps are not incompatible with other refinement processes such as GAC. They can be added in the procedure to work together with the constraint rules. For example, GAC can be added after Step II to further refine clusters.

6.5.3 Computational Complexity

We informally address the complexity of our approach. The algorithm in Step I performs a constraint check that normal canopy clustering doesn't have. This extra check does about $O(km^2)$ judgements where k is the number of clusters and m is the average size of each cluster. In the setting of duplicate detection, the size of each cluster usually is not very big ($k \gg m$). The complexity of cluster adjustments in Step

```
Input: Set of instances $D = \{d_1, d_2, \cdots, d_N\}$;
    Set of clusters $C_2$ generated from Step II.
Output: Set of duplicate pairs $P_1$.
Begin
  $P_1 = \emptyset$;
  for each $c_i \in C_2$ do
    for any $d_x, d_y \in c_i (x \neq y)$ do
      // primary similarity and derived similarity
      if $d_x \rightarrow d_m$ and $d_y \rightarrow d_n$ and $\exists c_j \in C_2$, $c_j(d_m, d_n)$
        $P_1 = P_1 + (d_x, d_y) + (d_m, d_n)$;
      end if
      // reinforced similarity
      if $d_x \rightarrow d_m$ and $d_n \rightarrow d_y$ and $\exists c_j \in C_2$, $c_j(d_m, d_n)$
        $P_1 = P_1 + (d_x, d_y) + (d_m, d_n)$;
      end if
    end for
  end for
  Output $P_1$;
End
```

Figure 6.5: The algorithm of Step III

II depends on the number of primary instances (p) and the average size of derived instances a primary instance has (q), which is about $O(pq^2)$. Normally, $n > p \gg q$ where n is the number of all the instances. In Step III, the extraction of potential duplicate pairs out of each cluster performs at the complexity level of $O(km^2 + km^2q^2)$ if we include the checking for the derived instances. The complexity in Step IV depends on the implementation. Our simple implementation operates at $O(p^2q^2)$. After all, it should be noted that all the above operations (checking, adjusting, extracting) don't involve very expensive computations. In fact, our experiments reveal that a lot of time is spent in computing the similarity between instances.

```
Input:
  Set of instances $D = \{d_1, d_2, \cdots, d_N\}$;
  Set of clusters $C_2$ generated from Step II.
  Set of Pairs $P_1$ generated from Step III.
Output:
  Set of duplicate pairs $P_2$.
Begin
  $P_2 = \emptyset$;
  for any $d_x, d_y \in D$ such that $x \neq y, d_x \not\approx d_y$ do
    $P_{tmp} = \emptyset$;
    while $\exists d_m, d_n, c$ such that
    $d_x \rightarrow d_m, d_y \rightarrow d_n, c \in C_2, c(d_m, d_n)$ do
      $P_{tmp} = P_{tmp} + (d_m, d_n)$;
    end while
    if $|P_{tmp}| > 1$ then $P_2 = P_2 + P_{tmp}$;
  end for
  $P_2 = P_2 + P_1$;
  Output $P_2$;
End
```

Figure 6.6: The algorithm of Step IV

6.6 Experiments

There exist some commonly used data sets for duplicate detection experiments, but data instances in these sets don't have many types and in-between relations. And mostly they are presented from one unified perspective, which doesn't represent well the real world situations of relational data such as the ontology data over the web. Therefore, we collected data from different sources to build the data set for our experiments. The data set is mainly about papers, authors, conferences/journals, publishers and their relations. Such data is collected from DBLP web site (http://dblp.uni-trier.de) and authors' home pages. These data instances are converted into a working format (i.e., OWL) but types, relations and original content values are preserved. Manual labeling work is done to identify the true duplicates among the data for the purpose of evaluation of approaches in the experiments. Totally, there are 278 data instances in the data set referring to 164 unique entities. The size may not be so big, but duplicate detection in it may not be easy since there are a certain amount of different instances with very high similarity, for example, different papers within same research fields, or different authors that have same/similar names. The distribution of duplicates is not uniform. About two-third of instances have one or two references to their corresponding entities. The most duplicated entity has 13 occurrences.

Same as [109], we use standard metrics in information retrieval to evaluate the performance of clustering approaches for duplicate detection. They are precision, recall and F measure. Precision is defined as the fraction of correct duplicate predictions among all pairs of instances that fall in the same resulting cluster. Recall is defined as the fraction of correct duplicate predictions among all pairs of instances that fall in the same original real duplicate cluster. F measure is the harmonic average of precision and recall.

We evaluate our approach in comparison with the canopy-based greedy agglomerative clustering approach (CB+GAC) [109]. CB+GAC also performs canopy clustering first but with no constraints. It then refine each canopy cluster using GAC: initialize each instance in the canopy to be a cluster, compute the similarity between all pairs of these clusters, sort the similarity score from highest to lowest, and repeatedly merge the two most similar clusters until clusters reach to a certain number. Table 6.1 shows the evaluation results of different approaches. The two threshold parameters for canopy clustering in this evaluation are set as $T_{tight} = 0.5$ and $T_{loose} = 0.35$, which are obtained through a tuning on a sampled data set. The number of clusters is then automatically determined by the two parameters. In the table, "CB+GAC" is the general clustering approach we have just discussed. "Step 12" is the approach that only performs Step I and Step II (refer to Section 6.5.2) and then returns the resulting clusters. "Step 12+GAC" is the approach that performs GAC after step I and step II. "Step 1234" obviously is the approach that performs all the steps to impose all the constraints we've defined on the clusters. From the table, we can see that by incorporating constraint rules, the overall F measure improves along with the precision. In particular, when all the constraints are applied, the precision increases up to 20%, which indicates that our approach can predict duplicate with very high accuracy.

Table 6.1: Performance of different approaches of duplicate detection

Approach	Precision	Recall	F score
CB+GAC	0.717	0.806	0.759
Step 12	0.728	0.877	0.796
Step 12+GAC	0.784	0.817	0.800
Step 1234	0.921	0.721	0.809

Figure 6.7 shows the sensitiveness of precision of different approaches to the loose similarity threshold (T_{loose}) in the canopy clustering. Since in our approach some constraint rules are used to extract duplicate pairs out of working clusters, the quality of the initial canopy clustering may affect the performance. That is, when T_{loose} becomes more loose, each canopy cluster may have more false duplicates, which might affect the performance of those constraint rules used for duplicate extraction. The trend of dropping precision while T_{loose} decreases is well revealed in approach "Step 12". However, the dropping trend of approach "Step 1234" is slightly better than that of "Step 12", which means that constraint rules used in Step III and Step IV can tolerate noisy canopy clusters to certain degrees.

Table 6.2 shows the precision of detecting duplicated pairs in different steps in our approach. This can be used to roughly estimate contributions of different constraint rules as they are implemented in different steps. The evaluation on our data set shows that the main contribution to the improved precision is made in Step III, where constraint rules of "primary similarity", "derived similarity" and "reinforced similarity" are imposed.

Table 6.2: Precision of detection of duplicated pairs in different steps

	Step I	Step II	Step III	Step IV
Precision	0.650	0.682	0.881	0.888

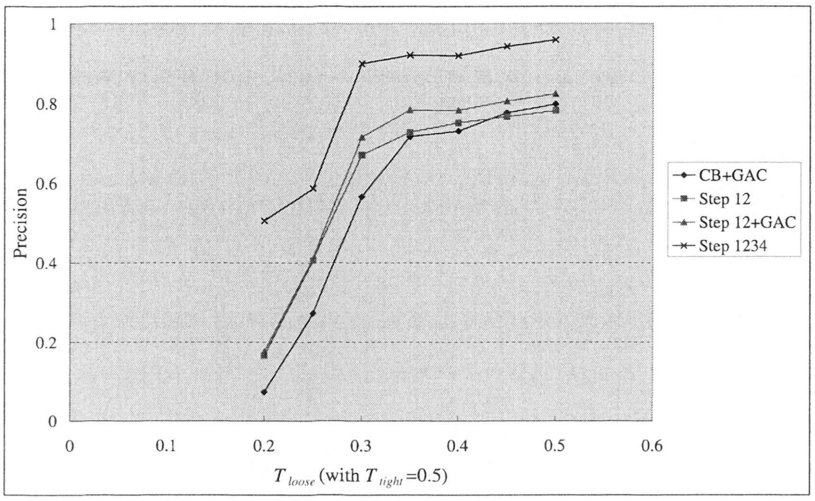


Figure 6.7: The sensitiveness of precision to T_{loose} for different approaches of duplicate detection

6.7 Summary

This chapter tackles the problem of duplications among the ontology data that happens during the data integration tasks. It shows several scenarios that demonstrate the significance of doing research on this issue. Then the characteristics of the ontology data have been analyzed from the perspective of duplicate detection. Based on these characteristics, we have defined constraint rules, which are implemented and incorporated in our cluster-based approach. Experiments show that our approach performs well with improved accuracy in term of precision and recall. Experimental evaluations also reveal that the use of constraint rules increases the precision of duplicate detection for ontology data with multiple perspectives.

Chapter 7

Ontology Data Matching through Learning Adaptive Metrics

7.1 Introduction

While Chapter 6 shows our constrained clustering method for ontology data matching, this chapter will discuss another method that matches ontology data through learning adaptive similarity metrics. In addition, a proposed peer-to-peer framework that uses this method for ontology data integration is also discussed.

Both supervised data matching methods (which the method discussed in this chapter belongs to) and unsupervised data matching methods (which the method discussed in Chapter 6 belongs to) can be used to yield satisfactory results if they fit the concrete problem settings well. For example, unsupervised data matching methods can do a good job when no labeled/tagged data are available for the matching tasks as they don't need a training process. On the other hand, supervised matching methods may do an even better job when a certain amount of labeled/tagged data do exist. One more benefit of the supervised matching methods is that the work of selecting proper thresholds or other parameters would be reduced to a certain degree thanks to the training mechanism. Therefore, the constrained clustering method discussed in Chapter 6 can be applied to the matching tasks when the raw data are not labeled/tagged, whereas the method discussed in this chapter is suitable for the situation when a certain amount of data have been labeled/tagged.

And it is also possible to make the two types of methods work together. For example, first the unsupervised matching method is performed over the raw ontology data that don't have any labels/tags. The initial results from the method can be reviewed by the user, especially those results that fall near the boundary. Given initial results and the feedback from the user, a set of labeled/tagged data can be created so that the supervised methods can be used to perform ontology data matching.

The method proposed in this chapter belongs to the class of supervised data matching methods. But different from other methods in this class [173], our method explores the features of ontology. We explain the benefits of using these features and discuss how they are captured in matching ontology data. Experiments are also conducted to verify that the exploration of ontology features does improve the accuracy of finding correct matching.

While the process of data warehousing often results in a single repository of integrated data for analysis [170], we may wonder it is desirable to create a single repository of ontology data over the large scale semantic web. The drawback is obvious as the dynamic semantic web can quickly outdate this single repository, making analytical results upon the repository less valuable. Therefore, we propose a flexible framework that is able to integrate relevant ontology data through a group of peers. Our particular matching method can be used in the framework to maintain a relation among the ontology data from different peers.

The rest of the chapter is organized as follows. Section 7.2 presents different similarity metrics use in ontology data matching including traditional similarity metrics and metrics that explore the ontology features. Section 7.3 discusses the use of the learning mechanism with support vector machines (SVM) for ontology data matching. Experiments with the method are presented in Section 7.4. Section 7.5 shows the peer-to-peer framework that supports ontology data matching as well as ontology data generation. Finally, Section 7.6 summarizes this chapter.

7.2 Similarity Metrics for Instance Matching

Instance matching can be performed by checking similarities between instances. If the similarity degree of two instances reaches a certain level, then the two instances can be regarded as matched. There are two major traditional metrics of similarity between instances: string edit distance [70] and cosine similarity based on TF-IDF [137]. Our proposed method uses both of them. In addition, we develop new metrics by exploring ontology features.

7.2.1 String Edit Distance and TF-IDF

String edit distance (also known as Levenshtein distance) is used to compare string similarity at the character level. It is defined as the minimum cost of transforming one string into another by insertions, deletions, or substitutions. For two strings S_1 and S_2 with length m and n respectively, their string edit distance $SED(S_1, S_2)$ can be computed by the dynamic programming algorithm [70]. Let $D(i,j)$ be the string edit distance between substring $S_1[1\ldots i]$ and $S_2[1\ldots j]$, then we have the following:

$$D(i,j) = \begin{cases} i & j = 0 \\ j & i = 0 \\ min[D(i-1,j)+1, D(i,j-1)+1, & \\ \qquad\qquad D(i-1,j-1)+c(i,j)] & other \end{cases} \tag{7.2.1}$$

and

$$SED(S_1, S_2) = D(m,n) \tag{7.2.2}$$

where cost $c(i,j) = 1$ for $S_1[i] \neq S_2[j]$ and $c(i,j) = 0$ for $S_1[i] = S_2[j]$.

This string edit distance is very effective in detecting typo problems. However, this measurement may not sensible when entire word(s) is inserted or added for comparison. In [120] the basic method is extended to measure gaps (contiguous mismatched characters) more reasonably when aligning the two strings for comparison. The computation of this extended string edit distance is still based on dynamic programming, but more flexible cost model is used and more matrices are required [18, 70].

Unlike string edit distance methods that compute the distances of string at character level, TF-IDF based methods use a vector space model [12], treating strings as "bag of tokens" and ignoring the sequential order of tokens in the strings. An instance that consists of one or more strings can be viewed as a virtual document containing a bag of string tokens. If there are N instances, then the corresponding N virtual documents form a virtual corpus, which may finally have a vocabulary of n distinct tokens. A sparse n-dimensional token vector V_i can be derived from the i-th virtual document with each element $v_{i,j}$ having TF-IDF value computed as follows:

$$v_{i,j} = TF_{i,j} \times \log \frac{N}{DF_j} \tag{7.2.3}$$

where $TF_{i,j}$ is the frequency of token t_j in the i-th document and DF_j is the frequency of the document that contains the token t_j.

For two instances with S and T as their derived token vectors respectively, the similarity between them is computed as normalized dot product between their corresponding token vectors.

$$SIM(S,T) = \frac{\sum_{j=1}^{n} s_j \cdot t_j}{\|S\| \cdot \|T\|}. \tag{7.2.4}$$

7.2.2 Exploring Ontology Features for New Metrics

The use of ontology enables us to enhance the method that only computes string similarities between ontology instances. Two ontology features are used for developing new similarity metrics .

The first feature is the ontology hierarchy. Given an ontology schema, we can compute the subsumption relations between the concepts in it using a specific reasoner. Then hierarchy of the concepts can be constructed, which allows us to explore the "concept-level similarity" of instances. Because ontology data are contributed by different information sources separately, the quality and the focus of completeness of the data vary. It can not be guaranteed that the instances referring to the same real world entity are identified exactly with the same concept by different information sources.

However, their concepts should have some relations according to their common ontology schema. For example, if two instances from different information sources are identified as instances of concept "`Student`" and "`GraduateStudent`" respectively, then they are more likely to be the same than two instances with one identified as "`Student`" and another as "`Professor`", given the two pairs have the same similarity degree measured by string edit distance or TF-IDF. In addition, we are able to define disjoint concepts in the ontology schema. If "`Student`" and "`Professor`" (both are sub concept of "`Person`") are defined as disjoint concepts, and we assume that each information source is aware of this assertion when contributing ontology data, then a student instance could never be matched with a professor instance, even they have very high string-based similarity.

We use "concept distance" to measure concept-level similarity. Suppose two instances i, j are of concept A and B respectively. This can be denoted as $A(i)$, $B(j)$. Concept distance between i and j, denoted by $CD(i,j)$, is computed as follows:

$$CD(i,j) = \begin{cases} 0 & A \equiv B, \\ PT(A,B) & A \sqsubseteq B \text{ or } B \sqsubseteq A, \\ PT(A,B) + P & A \not\sqsubseteq B,\ B \not\sqsubseteq A, \\ \infty & A \sqcap B = \perp . \end{cases} \tag{7.2.5}$$

where $PT(A,B)$ means the length of the path between concept A and B according to the computed concept hierarchical tree; P is a penalty item, always given a positive number. Since this measurement will be adjusted through the learning from labeled data, it is not very significant to choose the value of P as long as it is a positive number within a given range (i.e.,1-10). In our experiments, $P = 2$. For ∞, we use a big enough positive number to represent it in the implementation. Intuitively, if the concept distance of two instances are bigger, the likeness of being same would be less.

We also examine object properties of instances to compute "context similarity". Object properties allow users to create specific relations between instances. Usually, before an object property is employed to link instances with semantic relations, it is defined between concepts with an option to specify its cardinality constraints.

In addition, an object property can have an inverse object property, which allows more flexible ways of describing ontology data. Using inverse object properties could be very common among different information sources. For example, a publisher may tend to describe publications using a property "`author`" to relate them to their author instances, while a professor may choose to use the inverse property "`authorOf`" to link his own instance with his publication.

By reasoning on the inverse properties and checking the cardinalities on them, we can compute the context similarity between instances. For instance a and b, their context similarity in terms of the object property r, denoted by $CS_r(a,b)$, is computed as follows:

$$CS_r(a,b) = \frac{\sum_{\forall m, r(a,m)} \sum_{\forall n, r(b,n)} SIM(m,n)}{|\{m : r(a,m)\}| \times |\{n : r(b,n)\}|} \tag{7.2.6}$$

Equation 7.2.6 computes the average vector based similarity between the instances related to a and instances related to b via the object property r. The computational complexity of it is $O(M \times N)$ where M and N are the total number of instances related to a and b via r respectively. Obviously, this computation is very expensive if both a and b have a lot of r-related instances. To reduce the computational cost, we use an alternative method in the implementation by merging M (or N) instances into *one* virtual document for SIM computing. This alternative method only requires one SIM computation. Though the result might be different from the result computed via Equation 7.2.6, it still reflects the average similarity between the M and N instances.

7.3 Adaptive Metrics by the Learning Mechanism

So far, we have explored three sets of features to measure the similarity between instances: features based on word-level string similarity (SIM); features based on character-level string similarity (SED); and features related to ontology (CD and CS). The measurements based on these features are not unified. For example, SIM and CS measure similarity in different directions compared with SED and CD. In addition, these measurements are not normalized and how well each of them contributes to the final similarity between instances are not clearly known.

To tackle the issue of integrating different measurements, we employ a machine learning approach. We create a set of matched instance pairs with positive labels and a set of non-matched instance pairs with negative labels. A binary classifier is trained by using different similarity measurements as features from the two pair sets. This binary classifier then acts as a paring function [33] $h(a, b)$, taking a pair of instances a, b as input and generating decision values as output. If it generates positive values, the two input instances are regarded as matched; otherwise, unmatched.

Based on the discussion in [18], we choose support vector machine (SVM) [157] as the classification model in this study. SVM is able to learn from small training sets of high-dimensional data with satisfactory precision. Therefore, we create a feature vector of an instance pair using the separate property similarity measures for SED and SIM instead of their overall similarity measures. For a pair of instances a, b, its feature vector is composed as follows:

$$\begin{aligned}\mathbf{p}(a,b) \quad = \quad & [SIM_{d_1}(a,b), \ldots, SIM_{d_m}(a,b), \\ & SED_{d_1}(a,b), \ldots, SED_{d_m}(a,b), \\ & CS_{o_1}(a,b), \ldots, CS_{o_n}(a,b), \\ & CD(a,b)]. \end{aligned} \tag{7.3.1}$$

Although using separate property features increases the dimensions of the feature vectors greatly, the learned classifier can implicitly reflect the weight of different properties based on the training set, relying on more informative properties to make classification decisions.

SVM classifiers provide the following classification function:

$$f(\mathbf{q}) = \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{p_i}, \mathbf{q}) + b \tag{7.3.2}$$

where $K(\mathbf{p}, \mathbf{q})$ is a kernel function used for mapping features into different spaces, α_i

is the Lagrangian coefficient of the i-th training instance pair, $y_i \in \{-1, +1\}$ is the label of the training instance pair. In this function, α_i, b are obtained by computing the solution to the quadratic programming problem derived from maximization of the hyperplane margin between positive training samples and negative training samples. Given a test instance pair $\mathbf{q}$, we regard $\mathbf{q}$ as a matched pair if $f(\mathbf{q}) > 0$. Besides this, as $f(\mathbf{q})$ indicates the distance of $\mathbf{q}$ from the optimal hyperplane, we can use this value to evaluate the confidence level of the pair being matched [18]. That is, if $f(\mathbf{r}) > f(\mathbf{q})$, then $\mathbf{r}$ is more likely to be a matched pair than $\mathbf{q}$.

7.4 Experiments

This section describes the experiments we've conducted to test the effectiveness of the proposed method for ontology data matching. It shows the way the experimental data is collected [1], discusses the methodology used in the experiments and analyzes the experimental results.

7.4.1 Data Collection

The ontology schema (TBox) we used in the experiments is derived from the one created by [69]. The original ontology schema consists of a well developed hierarchy of concepts in the university domain (such as Professors, Publications, Students, and etc). We extend it by asserting a few new concepts and properties so that it can sufficiently annotate real world data.

The right ontology data off the shelf over the web are very rare at this stage. For our experimenting domain, data related to the university ontology schema are mostly published in other formats such as unstructured/semi-structured html pages or bibtex files. Therefore, wrappers are created to extract the desired data out of those related files over the web and then data conversion is performed to convert them into ontology

[1]Actually, this data collection process is based on the one described in Chapter 6. But this time, more data are collected.

Table 7.1: Match distribution of instances in the ontology data set

match size	number of match	number of instances
1	317	317
2	36	72
3	6	18
4	2	8
5	2	10
6	1	6
9	1	9
13	1	13
		Total: 453

data aligned with the ontology schema. Those original files are gathered from different web sites so that decentralization of data is achieved. Data focused on publication descriptions are collected from the DBLP web site (http://dblp.uni-trier.de). Data that describe academics, their publications, research interests, and etc are gathered from their personal home pages. Department-maintained web pages that describe their staff members' information are collected and processed as well.

After these data are processed and converted into ontology data format, we examine them manually to find out which instances are matched with each other and label them. Totally there are 453 instances in the collected data set. Table 7.1 lists the instance matching statistics of the data set.

7.4.2 Methodology and Results

From the labeled instance data set we generate a set of instance pairs using the canopy clustering method [109]. Because the canopy clustering is not very accurate though it runs fast, the resulting pair set contains not only the correctly matched pairs but also pairs that are mismatched. In other words, the resulting pair set contains both positive samples (matched pairs) and negative samples (mismatched pairs), ideal for training/testing a binary SVM classifier. The SVM^{light} package [90], one implementation of SVM classifier, is used to learn instance matching (as discussed in Section 7.3) with radial basis function as kernel function. For one experimental trial we split the pair set into two folds randomly, one for training, the other for testing and then reverse. Totally 20 random trials are conducted and the final results are obtained by averaging all these trials.

The results are reported in traditional measurement as *precision*, *recall* and *F measure* (or F1 measure in some literatures) [12]. Precision is defined as the ratio of the number of correctly identified matched pairs to the number of identified matched pairs, while recall is defined as the ratio of the number of correctly identified matched pairs to the total number of matched pairs in the data set. F measure is defined as the harmonic mean of precision and recall. When recall increases as more pairs with lower confidence of similarity are treated as matched, precision usually decreases. To generate an overall view, precision/recall curve is given where precision is plotted at eleven different recall levels.

We evaluate the effectiveness of incorporating ontology features into the matching problem by comparing it with the methods that use traditional features only (TF-IDF based similarity and String edit distance). Figure 7.1 illustrates the results of different methods in the form of precision/recall curve. The method that incorporates ontology features, denoted by "SIM+SED+ONTO", has higher precision than other methods at almost all the recall levels. This advantage becomes less at very high recall levels (e.g. 0.9, 1.0) compared with methods that use TF-IDF/SED measures ("SIM", "SED", "SIM+SED"). This is because TF-IDF or SED based measurement only cares about string-based similarity between the instances. Some instances have very similar string representations but actually refer to different real world entities, for example, different persons with similar/same names. Methods based on TF-IDF or SED based measurement will then misclassify them as matched pairs with very high confidence, which results in a relatively low precision at a modest recall level. On the contrary, the ontology feature enhanced method is able to capture the differences of those instances by examining their ontology features, which leads to the decrease of similarity measures for those instances. Therefore the precision is accordingly boosted.

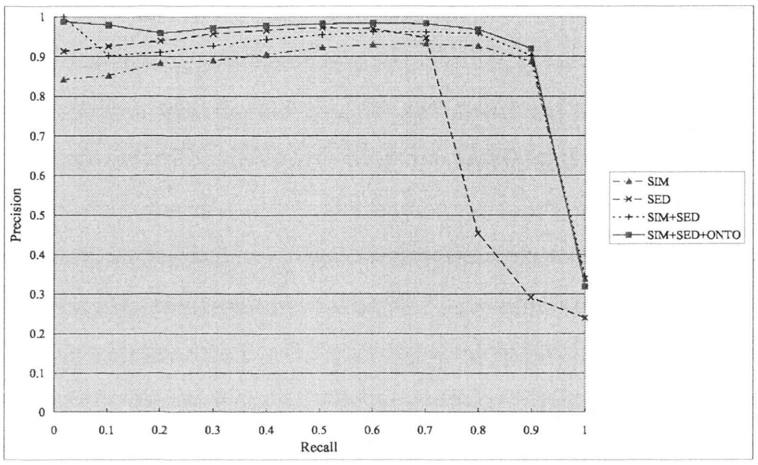


Figure 7.1: Recall/precision curves of different methods

Table 7.2 presents the results of different methods in F measure. This F measure of each method is obtained by averaging the maximum F measure of each trial. It shows that using more features yields better results. By incorporating ontology features, we achieves the highest F measure score. Although the improvement between "SIM+SED+ONTO" and "SIM+SED" is less than that between "SIM+SED" and "SIM" (or "SED"), our one-tailed paired t-test shows that the result is significant with $p < 0.05$.

Table 7.2: Averaged Maximum F measure of different methods

Methods	Averaged Max F
SIM	0.9081
SED	0.8359
SIM+SED	0.9240
SIM+SED+ONTO	**0.9365**

Since a large proportion of the instances in the data set belong to the class `Person` and `Publication` (including their sub classes), we also examine the performances on these instances separately. Figure 7.2 shows the results in F measure. The method that explores ontology features outperforms other methods on the instances of class `Publication`. For instances of class `Person`, its F measure score is slightly less than the method that uses "SIM" only. The reason may be that many instances of the class `Person` in our data set are described in a similar pattern with little ontology enhanced explanation, which makes the "SIM" feature sufficient enough for classification. However, we believe that in a dynamic distributed environment, the instances contributed from different peers are more diverse and with different perspectives, as those of the class `Publication`. Therefore, the use of ontology features will help increase the performance of ontology data matching.

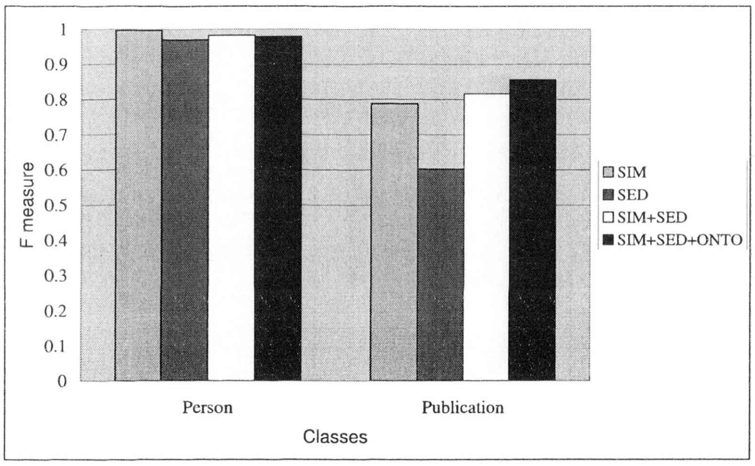


Figure 7.2: F measure scores of different methods on two major classes: `Person` and `Publication`

Generally, the experiments confirm that the accuracy of ontology data matching can be improved by our method that explores the ontology features.

7.5 A Peer-to-peer Framework for Ontology Data Integration

To make the ontology data available on the web and able to be effectively integrated by the above discussed method, we propose a corresponding peer-to-peer framework. In this framework, super peer topology [122], one of the peer-to-peer architectures [171], is employed to fit the nature of decentralization of peer data management.

Actually, the framework is also designed with the function of ontology data generation that has been discussed in Chapter 4. So it supports both the generation and the integration of ontology data.

7.5.1 Overview of the Framework

We first give an brief overview of the framework that supports the generation and matching of ontology data.

The framework uses the super peer topology [122] for the peer-to-peer architecture. Peer-to-peer architectures have evolved into two rough categories according to their degree of centralization [171]: pure peer-to-peer and hybrid peer-to-peer. Super peer topology falls into the hybrid one, which has one or more central servers as super peers. Different from traditional client/server architecture, it shifts part of the tasks of the server (super peer) to the clients (ordinary peers). For example, when processing a query, a super peer can simply tell the query issuer from which peers they can get potential query results instead of returning the complete query results directly. In addition, there can be several super peers with a typical hierarchy in a system so that server tasks can be distributed among them with some predefined policies to achieve high performance.

Accordingly, as shown in Figure 7.3, the framework introduces two types of peers: super peer and ordinary peer (or just denoted as peer). The super peer acts as a coordinator for peers connected to it. It hosts the backbone ontology schema used

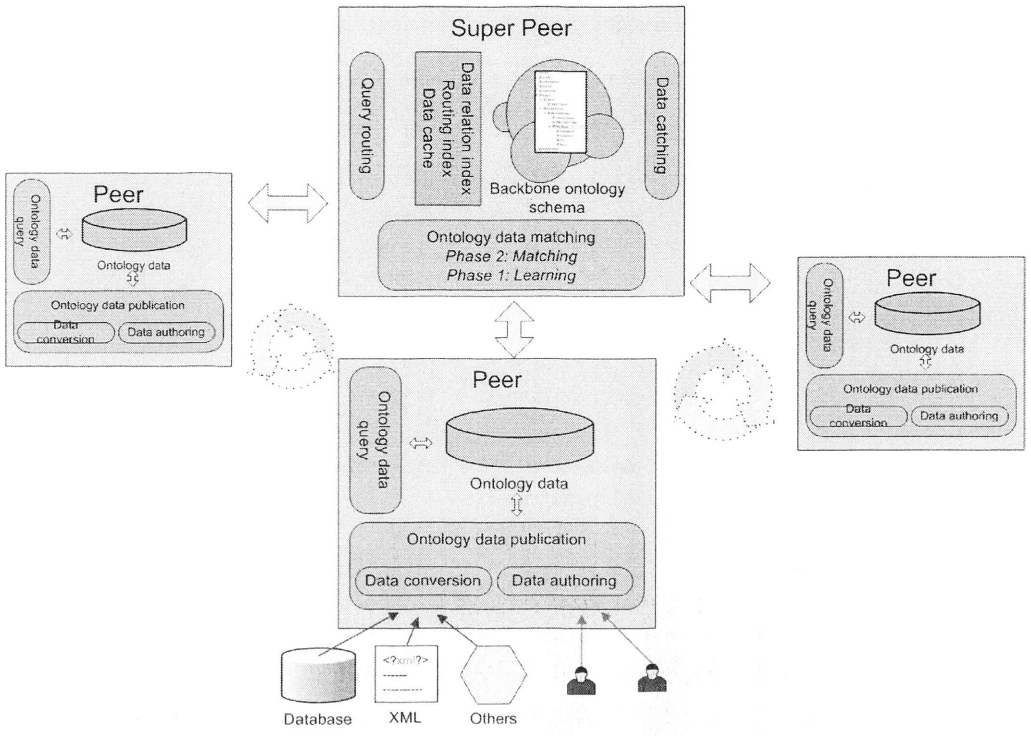


Figure 7.3: The framework that supports ontology generation and matching with a design of a peer-to-peer architecture

for the generation and integration of ontology data. Additional information is stored at the super peer to make it work as the coordinator sufficiently. *Routing index* helps the super peer identify which peer contributes what kind of data. *Data relation index* maintains semantic relations created between ontology data segments contributed by different peers. *Data cache* is used to cache data from peers who allow their data to be cached for better performance. To create those kinds of information and then to use them in turn, the super peer offers relevant functions or services: *data caching*, *query routing*, and *ontology data matching*. For example, the function of *ontology data matching* creates and updates the data relation index by the matching method. It will be discussed in the following sub section. In contrast, peers are more light-weighted, offering functions that support ontology data generation (*ontology data publication*) and query (*ontology data query*). As the figure illustrates, the function of ontology data publication consists of data conversion and data authoring. The methods that can be used to implement this function have been discussed in Chapter 4.

7.5.2 Ontology Data Matching in the Framework

The ontology data matching process is designed to deal with the problem of implicit relations among the data from different peers, i.e., the data-level integration in contrast to schema-level integration (refer to Section 2.5.3). Obviously, the task of this process is to match the instances from different peers, making their implicit relations explicit.

Accordingly, this process consists of two phases: a learning phase and an matching phase.

The **learning phase** involves the training of the SVM classifier from matched and unmatched instances. First, a certain amount of initial data are gathered by super peer from different peers. This data set should contain a portion of matched instances. These matched instances are not discovered and specified initially, while training an SVM classifier requires both specified matched instances and unmatched instances as positive and negative samples. Therefore, these initial data should be checked and tagged manually for the training. An initial similarity checking and sorting process based on selected instances properties (known as *blocking* [124][123] discussed in Section 2.5.3) is performed to make the manual tagging easier. Currently, we are incorporating the constrained clustering method discussed in Chapter 6 to assist this process. After all these initial data are tagged and the matched and unmatched instance pairs are created, the classifier is trained as discussed in Section 7.3.

The **matching phase** allows the super peer to match the instances from the peers when they contribute their own data. When a peer has some data published, the super peer will be notified. Instances in those data will be sent to the super peer for an initial check upon its request. The initial check searches potentially matched instances that are previously indexed for the new instances through an inverted index. If no instances are found for the new instances or the found instances have very low hits, these new instances will be ignored. This initial check screens off a number of instances. For those instances with potentially matched instances found, instance pairs are created as the input of the trained classifier. Then the trained SVM classifier is used to determine if the instance pairs are matched pairs with Equation 7.3.2. After matched instance pairs are found through this classification process, the index storing data relation information among peers is updated by adding information about these pairs. This index reveals the semantic relations of the data across the peers.

As peers that contribute ontology data often have different perspective and different emphasis on data completeness, once those data are semantically linked with each other by instance matching, incompleteness of one source can be compensated by other sources. This then enables people to acquire information or even knowledge from across the sources.

While the peer-to-peer framework seems promising in integrating ontology data, some limitations should be mentioned to avoid inappropriate uses and implementations. First, the use of a super peer topology casts more responsibilities on the super peer than on ordinary peers. This could cause problems when the super peer is not implemented in a robust and reliable way. Adding more super peers into the network may reduce this risk, but this introduces more complexities, which need to be addressed properly. Second, when the ontology data grow to a large amount, the current matching method in the framework may not be efficient enough to handle them. Some optimizations are required to address the efficiency issue.

7.6 Summary

This chapter continues to tackle the issue of ontology data matching. Different from last chapter (Chapter 6), it proposes unsupervised matching method. When some labeled/tagged ontology data are available, this method is able to learn the adaptive similarity metrics from them for more effective matching. Particularly, unlike traditional data matching methods, it explores ontology features for new metrics. The introduction of new metrics improves the matching performance in terms of accuracy shown in our experiments. Finally, based on this matching method, we propose a flexible peer-to-peer framework to address the issue of ontology data publication and integration in the distributed semantic web environment. We expect this framework may allow us to view the semantic web as a very large and ever evolving data warehouse where further intelligent analysis can be performed to deliver high level knowledge.

Chapter 8

Conclusions and Future Work

This chapter draws conclusions from the study presented in this thesis and indicates some main directions for future work.

8.1 Conclusions

This thesis studies three main issues that are related to ontology-based web data integration. They are: (1) How to develop the domain ontology schema(s) used for the integration; (2) How to generate ontology data (mostly on the web) for domain ontology schema if the data are not in the right format and to manage the web ontology data in an appropriate way; (3) How to improve the quality of integrated ontology data by reducing duplications and increasing completeness and certainty. The significance of the studies has been discussed and related work on these research issues have been analyzed. Particularly, a set of new methods have been developed to deal with these issues from different angles, which can be concluded as follows:

- A key information mining method is proposed and its application to domain ontology schema generation is presented. Further more, a prototype system is developed based on the method and its ability to help develop ontology schemas in given domains is shown.

- Two methods for web ontology data generation are proposed for the purpose of facilitating the generation of web ontology data for given domains. One method leads to a web-based system that enables ordinary users to publish domain ontology data with ease. In addition, an ontology enhanced model is proposed to enforce a flexible management mechanism on the system so that users can manage the web ontology data collaboratively but also securely to a certain degree.

- Two methods are developed to perform ontology data matching for the improvement of ontology data quality when integration of them is needed. One method uses the clustering approach. It makes use of the relational nature of the ontology data and captures different situations of matching, resulting in an improvement of performance compared with the traditional clustering method. The other method goes further by using a learning mechanism to make the matching more adaptive. New features are developed for training matching classifier by exploring particular characteristics of ontology data. It also achieves better performance than those with only ordinary features. Further more, a peer-to-peer framework is proposed to integrate available ontology data from different peers. The mentioned matching methods are designed to discover matchings among the ontology data across the peers so that the overall quality of the ontology data in the framework can be improved.

The above methods have been evaluated with related experiments, which have demonstrated their effectiveness, efficiency and validity for the given situations. These methods also lead to the development of prototype systems, whose effectiveness is shown through the discussion of the functions of the resulting systems.

These methods can be applied to tackle the issues related to ontology-based web data integration in practice. Particularly, the prototype systems can be further developed to have applications in real world domains, as discussed in the next section.

8.2 Future Work

Although we have achieved some successes in tackling the issues related to ontology-based data integration, there exist more research tasks that are worth taking in the future work. In another sense, the issues in this research area are so significant and challenging and the topics are so attractive that a further pursuit is worthwhile.

The future work is discussed in the following several subsections respectively based on the current studies presented in this thesis.

8.2.1 More Functions and Applications with the KIM Method

While current prototype system with the KIM method is able to extract concepts and classes with a taxonomy structure for developing ontology schemas for given domains, the functions of it can be improved and enriched by adding other data mining techniques. For example, related web pages can be analyzed and added as instances of the concepts for further processing and query. This could result in news methods for ontology data generation. By analyzing web pages relating to one class (concept), potential attributes of the related concept can be identified for sophisticated domain ontology schema generation tasks, for example the refinement of properties of particular concepts.

While the above further study falls in the research direction of ontology generation for given domains, it is also possible to apply the KIM method to other research fields in the future. For example, since key information extracted from most web sites may contain meaningful information such as catalogue classification information, applications related to web page classification or catalogue integration can benefit from it. By treating key information as labels, quality training samples can be created to build classification or integration models. Another further study is to incorporate the KIM method into the construction of particular classification or integration models to reduce the cost of labeling large training samples.

8.2.2 Improving Web Ontology Data Generation and Management

The *robinet* system proposed for web ontology data generation and management can be further improved by adding more features.

First, interaction of the web-based system with ordinary users will be further improved to offer better usability. New technologies such as web 2.0 related techniques can be utilized to make such improvements.

Second, more advanced functions, such as sophisticated search and query functions, are to be designed so that computers or end users can make better use of the generated ontology data.

Third, as for web ontology data management, further studies on rule engineering and management, efficiency of rule processing will be carried out. System performances in real world situations will be evaluated as well.

8.2.3 Improving Ontology Data Matching Methods

Since we have proposed two methods for ontology data matching, we discuss the future work of them respectively.

Constrained clustering. One of the further studies for this method is to design quantitative metrics to reflect characteristics of duplicated relational data. The ideal metrics will act as *soft* constraint rules. Thus, they are expected to be more accurate for different duplicate problems. Another further study is to conduct further experiments with larger data sets. Currently, we are keeping collecting data from different sources and converting and labeling them to build larger data sets. Besides the evaluation of accuracy on the large data sets, the efficiency of the approach will be formally evaluated. By evaluating and analyzing results from the experiments on the larger data sets, possible optimizations can be made to improve the performance of the method.

Matching instances by learning. One possible direction is to further explore the characteristics of ontology data so that new metrics can be proposed for training the matching method. Another possible direction is to study other learning mechanisms that can fit the problem settings of ontology data matching even better. For example, conditional random fields (CRFs) [150] could be a potential learning model for matching ontology data. The further study of adopting a new leaning mechanism involves the establishment of the learning model for the problem, possible customization and optimization according to the problem, and experimental evaluations of the new mechanism.

In addition to improving the two types of methods respectively, the study of combining them in a proper way for better performance is also desirable.

Further more, as these methods can be incorporated into the *robinet* system developed for web ontology data generation, it is possible to carry out the study of using feedbacks from users during users' interaction to improve the matching.

8.2.4 Towards Applications in Real World Domains and Challenges

The proposed peer-to-peer framework in the thesis actually tries to put our relevant studies (ontology data generation, integration and matching) together. While it enables users to publish ontology data, integrate them together, and relate them semantically by matching, it is possible to make applications of it to help tackle real world issues, potentially in the domains or areas such as semantic web, security informatics, medical data integration, and etc.

For example, its application in medical data integration may be roughly sketched as follows. First, an ontology schema that describes the medical domain is developed by referring to related standardization efforts such as the work of HL7 [1]. With this domain ontology schema ready, medical organizations such as hospitals, clinics or medical centers can load it into the *robinet* web-based system and use the system to

[1]http://www.hl7.org/

publish medical data (Chapter 4). The system needs to be configured and proper operations and rules need to be developed and deployed into the system to ensure that such operations on the medical data don't violate the privacy policy (Chapter 5). When each medical organization has used the *robinet* system to publish and host medical data, the peer-to-peer framework can be setup by treating each organization as a peer, thus making it possible to integrate and share the medical data between organizations. The matching methods (Chapters 6 and 7) help to link a patient's medical data from different organizations so that authorized doctors can have a clear view of the patient's medical conditions, therefore able to make informed decisions.

Furthermore, the framework can be further extended by adding more advanced and intelligent services. For example, developing advanced query and reasoning facilities on the integrated ontology data in the framework can be studied in the future. It is believed that the services and facilities built upon the framework will be more powerful since they can access multiple sources via the semantic relations created by the matching method. And the underlying ontology schemas allow them to perform with a degree of flexibility and intelligence. In the case of medical data integration, medical researchers could use these services and facilities to help gain a deeper insight into the nature of different diseases.

While the above sketch may look quite straightforward, we believe the implementation in the real world can still encounter a number of practical issues and challenges that require further studies. Some major issues and challenges may include the following: 1) The development and maintenance of practical ontology schemas for a large real world domain is even more difficult compared to developing ontology schemas used for particular applications that are focused on limited aspects. While a single method alone can hardly handle this task, a set of methods that are carefully selected and combined are more desirable. 2) When the volume of the dealt data become very huge, the issue of scalability needs to be proper addressed. For example, the matching methods need to be optimized or even redesigned to handle the matching and integration of large amount of ontology data. Details of implementations such as storage and indexing of such amount of data need to be considered carefully. 3) The issue of efficiency also needs to be addressed when the amount of client requests increase rapidly. 4) The ways of designing intelligent services that makes the most use of the ontology data deserve further studies as well.

Bibliography

[1] Eneko Agirre, Olatz Ansa, Eduard Hovy, and David Martínez. Enriching very large ontologies using the www. In *Proceedings of Workshop on Ontology Construction of the European Conference of AI (ECAI-2000)*, 2000. http://arxiv.org/pdf/cs.CL/0010026.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, 1993.

[3] Rakesh Agrawal and Ramakrishnan Srikant. On integrating catalogs. In *Proceedings of the tenth international conference on World Wide Web*, pages 603–612, Hong Kong, 2001. ACM Press.

[4] Enrique Alfonseca and Suresh Manandhar. Improving an ontology refinement method with hyponymy patterns. In *Proceedings of the third Conference of Language Resources and Evaluation (LREC-2002)*, 2002. http://citeseer.ist.psu.edu/alfonseca02improving.html.

[5] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services: Concepts, Architectures and Applications*. Springer, 2004.

[6] Rohit Ananthakrishna, Surajit Chaudhuri, and Venkatesh Ganti. Eliminating fuzzy duplicates in data warehouses. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB-2002)*, pages 586–597, 2002.

[7] Yigal Arens, Chun-Nan Hsu, and Craig A. Knoblock. Query processing in the sims information mediator. In *Readings in agents*, pages 82–90. Morgan Kaufmann Publishers Inc., 1997.

[8] Jose L. Arjona, Rafael Corchuelo, David Ruiz, and Miguel Toro. From wrapping to knowledge. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):310–323, 2007.

[9] Soren Auer. Powl - a web based platform for collaborative semantic web development. In *Proceedings of the First Workshop Scripting for the Semantic Web (SFSW'05)*, 2005. http://www.semanticscripting.org/SFSW2005/papers/Auer-Powl.pdf.

[10] Sören Auer, Sebastian Dietzold, and Thomas Riechert. Ontowiki - a tool for social, semantic collaboration. In *Proceedings of the 5th International Semantic Web Conferent (ISWC 2006)*, volume 4273 of *Lecture Notes in Computer Science*, pages 736–749. Springer, 2006.

[11] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The description logic handbook : theory, implementation, and applications.* Cambridge University Press, New York, 2002.

[12] R. Baeza-Yates and Berthier de Araaujo Neto Ribeiro. *Modern information retrieval.* Addison-Wesley Longman, 1999.

[13] R. Baxter, P. Christen, and T. Churches. A comparison of fast blocking methods for record linkage. In *Proceedings of ACM SIGKDD '03 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 25–27, 2003.

[14] T. Berners-Lee, R.T. Fielding, and L. Masinter. Uniform resource identifier (uri): Generic syntax. ietf rfp 3986 (standards track), internet eng. task force,. http://www.ietf.org/rfc/rfc3986.txt, 2005.

[15] Tim Berners-Lee. Semantic web - talk at xml2000 http://www.w3.org/2000/talks/1206-xml2k-tbl/slide10-0.html, 2000.

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

[17] Indrajit Bhattacharya and Lise Getoor. Entity resolution in graphs. In Lawrence B. Holder and Diane J. Cook, editors, *Mining Graph Data*, pages 311–344. Wiley, 2006.

[18] Mikhail Bilenko and Raymond J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48, New York, NY, USA, 2003. ACM Press.

[19] Scott Boag, Don Chamberlin, Mary F. Fernndez, Daniela Florescu, Jonathan Robie, and Jrme Simon. Xquery 1.0: An xml query language (http://www.w3.org/tr/xquery), January 2007.

[20] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and Franois Yergeau. Extensible markup language (xml) 1.0 (fourth edition) w3c recommendation. http://www.w3.org/tr/xml, August 2006.

[21] K.K. Breitman, M.A. Casanova, and W Truszkowski. *Semantic Web: Concepts, Technologies and Applications*. Springer, 2007.

[22] Dan Brickley and R.V. Guha. Rdf vocabulary description language 1.0: Rdf schema. w3c recommendation. http://www.w3.org/tr/2004/rec-rdf-schema-20040210, 2004.

[23] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web*, pages 107–117, 1998.

[24] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. A framework for ontology integration. In Isabel F. Cruz, Stefan Decker, Jérôme Euzenat, and Deborah L. McGuinness, editors, *The Emerging Semantic Web*, volume 75 of *Frontiers in Artificial Intelligence and Applications*. IOS press, 2001.

[25] Soumen Chakrabarti. *Mining the Web : discovering knowledge from hypertext data*. Morgan Kaufmann series in data management systems. Morgan Kaufmann, Amsterdam ; Boston, 2003.

[26] Hsinchun Chen. Intelligence and security informatics: information systems perspective. *Decision Support Systems*, 41:555–559, 2006.

[27] Yu Chen, Wei-Ying Ma, and Hong-Jiang Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *Proceedings of the 12th international conference on World Wide Web (WWW '03)*, pages 225–233, New York, NY, USA, 2003. ACM Press.

[28] Peter Christen and Tim Churches. Secure health data linkage and geocoding: Current approaches and research directions. In *Proceedings of the National e-Health Privacy and Security Symposium (ehPASS)*, 2006.

[29] Philipp Cimiano, Siegfried Handschuh, and Steffen Staab. Towards the self-annotating web. In *Proceedings of the 13th international conference on World Wide Web(WWW '04)*, pages 462–471, 2004.

[30] Philipp Cimiano, Andreas Hotho, and Steffen Staab. Comparing conceptual, divisive and agglomerative clustering for learning taxonomies from text. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI)*, pages 435–439, 2004.

[31] Philipp Cimiano, Andreas Hotho, and Steffen Staab. Learning concept hierarchies from text corpora using formal concept analysis. *Journal of Artificial Intelligence Research*, 24:305–339, 2005.

[32] Fabio Ciravegna, Sam Chapman, Alexiei Dingli, and Yorick Wilks. Learning to harvest information for the semantic web. In *Lecture Notes in Computer Science : The Semantic Web: Research and Applications (ESWS 2004)*, volume Volume 3053/2004, 2004.

[33] William W. Cohen and Jacob Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '02)*, pages 475–480, New York, NY, USA, 2002. ACM Press.

[34] Enrico Coiera. *A guide to health informatics*. Arnold Publication, 2nd edition, 2003.

[35] T. M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley series in telecommunications. Wiley, New York, 1991.

[36] Isabel F. Cruz, Huiyong Xiao, and Feihong Hsu. Peer-to-peer semantic integration of xml and rdf data sources. In *Proceedings of the Third International Workshop on Agents and Peer-to-Peer Computing (AP2PC 2004)*. 2004.

[37] Aron Culotta and Andrew McCallum. Joint deduplication of multiple record types in relational data. In *Proceedings of the Fourteenth Conference on Information and Knowledge Management (CIKM)*, 2005.

[38] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the web services web: an introduction to soap, wsdl, and uddi. *IEEE Internet Computing*, 6:86–93, 2002.

[39] Honghua Dai and Bamshad Mobasher. Using ontologies to discover domain-level web usage profiles. In *Proceedings of the Second Workshop on Semantic Web Mining, at the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'02)*, 2002.

[40] Honghua Dai and Bamshad Mobasher. Integrating semantic knowledge with web usage mining for personalization. In Anthony Scime, editor, *Web Mining: Applications and Techniques*. Idea Group Publishing, 2004.

[41] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In *Proceedings of International Workshop on Policies for Distributed Systems and Networks*, 2001.

[42] H. Davalcu, S. Vadrevu, S. Nagarajan, and I. V. Ramakrishnan. Ontominer: bootstrapping and populating ontologies from domain-specific web sites. *IEEE Intelligent Systems*, 18(5):24–33, 2003. Publisher: IEEE, USA.

[43] Kushal Dave, Steve Lawrence, and David M. Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *Proceedings of the twelfth international conference on World Wide Web*, pages 519–528. ACM Press, Budapest, Hungary, 2003.

[44] Sren Auer David Aumueller. Towards a semantic wiki experience desktop integration and interactivity in wiksar. In *Proceedings of the 1st Workshop on the Semantic Desktop in conjuction with the 4th International Semantic Web Conference*, 2005.

[45] Hasan Davulcu, Srinivas Vadrevu, and Saravanakumar Nagarajan. Ontominer: bootstrapping ontologies from overlapping domain specific web sites. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters (WWW Alt. '04)*, pages 500–501, New York, NY, USA, 2004. ACM Press.

[46] S. Decker, M. Erdmann, D. Fensel, and R. Studer, editors. *Ontobroker: ontology based access to distributed and semi-structured information.* Database Semantics. Semantic Issues in Multimedia Systems. IFIP TC2/WG2.6 Eighth Working Conference on Database Semantics (DS-8). Kluwer Academic Publishers, 1999.

[47] D. Dey, S. Sarkar, and P. De. A distance-based approach to entity reconciliation in heterogeneous databases. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):567–582, 2002.

[48] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. Semtag and seeker: bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the 12th international conference on World Wide Web (WWW '03)*, pages 178–186, New York, NY, USA, 2003. ACM Press.

[49] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. Swoogle: a search and metadata engine for the semantic web. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management (CIKM '04)*, pages 652–659, New York, NY, USA, 2004. ACM Press.

[50] Ying Ding and Foo Schubert. Ontology research and development. part 1 - a review of ontology generation. *Journal of Information Science*, 28(2):123–36, 2002.

[51] AnHai Doan, Pedro Domingos, and Alon Levy. Learning source descriptions for data integration. In *Proceedings of the Third International Workshop on the Web and Databases*, pages 81–86, 2000.

[52] AnHai Doan and Alon Y. Halevy. Semantic-integration research in the database community. *AI Mag.*, 26(1):83–94, 2005.

[53] Andreas Faatz and Ralf Steinmetz. Ontology enrichment with texts from the www. In *Proceedings of the 2nd Workshop of Semantic Web Mining in conjunction with ECML/PKDD-2002*, 2002.

[54] David Faure and Claire Nedellec. A corpus-based conceptual clustering method for verb frames and ontology acquisition. In *Proceedings of the LREC workshop on adapting lexical and corpus resources to sublanguages and applications*, 1998.

[55] C. Fellbaum. *WordNet: an Electronic Lexical Database.* MIT Press, 1998.

[56] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969.

[57] Dieter Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce.* Springer, 2004.

[58] Jochen Fischer, Zeno Gantner, Manuel Stritt, Steffen Rendle, and Lars Schmidt-Thieme. Ideas and improvements for semantic wikis. In *Proceedings of the 3rd European Semantic Web Conference (ESWC 2006)*, pages 650–663, 2006.

[59] Dayne Freitag and Nicholas Kushmerick. Boosted wrapper induction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 577–583. AAAI Press / The MIT Press, 2000.

[60] John H. Gennari, Mark A. Musen, Ray W. Fergerson, William E. Grosso, Monica Crubezy, Henrik Eriksson, Natalya F. Noy, and Samson W. Tu. The evolution of protege: an environment for knowledge-based systems development. *Int. J. Hum.-Comput. Stud.*, 58(1):89–123, 2003.

[61] Cheng Hian Goh. *Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems.* PhD thesis, MIT, 1997.

[62] Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological engineering : with examples from the areas of knowledge management, e-commerce and the semantic Web.* Springer, 2004.

[63] Asunción Gómez-Pérez and David Manzano-Macho. Deliverable 1.5: A survey of ontology learning methods and techniques. Technical report, Universidad Politécnica de Madrid, 2003.

[64] B. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *Proceedings of 12th International Conference on the World Wide Web (WWW2003)*, 2003.

[65] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, 1993.

[66] Lifang Gu, Rohan Baxter, Deanne Vickers, and Chris Rainsford. Record linkage: Current practice and future directions. Technical report, CSIRO Mathematical and Information Sciences, GPO Box 664, Canberra 2601, Australia, 2003.

[67] Nicola Guarino. Formal ontology and information systems. In *Proceedings of the First International Conference on Formal Ontology in Information Systems (FOIS'98)*, pages 3–15, 1998.

[68] S. Guha, N. Koudas, A. Marathe, and D Srivastava. Merging the results of approximate match operations. In *Proceedings of the 30th VLDB Conference*, pages 636–647, 2004.

[69] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. An evaluation of knowledge base systems for large owl datasets. In *Proceedings of the 3rd International Semantic Web Conference (ISWC 2004)*, pages 274–288, 2004.

[70] Dan Gusfield. *Algorithms on strings, trees, and sequences : computer science and computational biology*. Cambridge University Press, Cambridge [England] ; New York, 1997.

[71] L. M. Haas, E. T. Lin, and M. A. Roth. Data integration through database federation. *IBM Systems Journal*, 41(4):578–596, 2002.

[72] Udo Hahn and Klemens Schnattinger. Towards text knowledge engineering. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 524–531, 1998.

[73] Alon Y. Halevy. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001.

[74] Alon Y. Halevy, Zachary G. Ives, Peter Mork, and Igor Tatarinov. Piazza: data management infrastructure for semantic web applications. In *Proceedings of the 12th international conference on World Wide Web*, pages 556–567. ACM Press, Budapest, Hungary, 2003.

[75] A.Y. Halevy, Z.G. Ives, Jayant Madhavan, P. Mork, D. Suciu, and I. Tatarinov. The piazza peer data management system. *Knowledge and Data Engineering, IEEE Transactions on*, 16(7):787–798, 2004. TY - JOUR.

[76] J. Han and K. Chang. Data mining for web intelligence. *IEEE Computer*, 2002.

[77] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2000.

[78] Siegfried Handschuh, Steffen Staab, and Alexander Maedche. Cream: creating relational metadata with a component-based, ontology-driven annotation framework. In *Proceedings of the international conference on Knowledge capture*, pages 76–83. ACM Press, Victoria, British Columbia, Canada, 2001.

[79] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistic*, 1992.

[80] Marti A. Hearst. Automated discovery of wordnet relations. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database and Some of its Applications*, pages 132–152. MIT Press, 1998.

[81] J. Heflin, J. Hendler, and S. Luke. Shoe: A knowledge representation language for internet applications, 1999.

[82] Mauricio A. Hernandez and Salvatore J. Stolfo. The merge/purge problem for large databases. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data (SIGMOD '95)*, pages 127–138, New York, NY, USA, 1995. ACM Press.

[83] Ian Horrocks. Daml+oil: a description logic for the semantic web. *IEEE Data Eng. Bull.*, 25(1):4–9, 2002.

[84] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. Swrl: A semantic web rule language combining owl and ruleml, May 2004.

[85] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM Press, Seattle, WA, USA, 2004.

[86] Unicode Inc. What is unicode? [http://www.unicode.org/standard/whatisunicode.html], 2007.

[87] Angelo Di Iorio, Valentina Presutti, and Fabio Vitali. Wikifactory: a web ontology-based application for creating domain-oriented wikis. In *Proceedings of the 3rd European Semantic Web Conference (ESWC 2006)*, 2006.

[88] Jan Jannink. Thesaurus entry extraction from an on-line dictionary. In *Proceedings of Fusion '99*, 1999.

[89] A. D. Jhingran, N. Mattos, and H. Pirahesh. Information integration: A research agenda. *IBM Systems Journal*, 41(4):555–562, 2002.

[90] Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML '98)*, pages 137–142, London, UK, 1998. Springer-Verlag.

[91] Lalana Kagal. Rei: A policy language for the me-centric project. Technical report, HP Labs, September 2002. http://www.hpl.hp.com/techreports/2002/HPL-2002-270.html.

[92] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca-Grau, and James Hendler. Swoop: A 'web' ontology editing browser. *Journal of Web Semantics*, 4(2):144–153, June 2006.

[93] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, pages 741–843, 1995.

[94] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[95] R Kosala, M Bruynooghe, JV den Bussche, and H Blockeel. Information extraction from web documents based on local unranked tree automaton inference. In *Proceedings of Eighteenth International Joint Conference on Artificial Intelligence*. 2003.

[96] Markus Krotzsch and Max Volkel Denny Vrandecic. Wikipedia and the semantic web - the missing links. In *Proceedings of the 1st International Wikimedia Conference, Wikimania*, Aug 2005.

[97] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1-2):15–68, 2000.

[98] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *Proceedings of*

the Eighteenth International Conference on Machine Learning (ICML 2001), pages 282–289. Morgan Kaufmann, 2001.

[99] Mong Li Lee, Hongjun Lu, Tok Wang Ling, and Yee Teng Ko. Cleansing data for mining and warehousing. In *Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA-99)*, Florence, Italy, August 1999.

[100] Maurizio Lenzerini. Data integration: a theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '02)*, 2002.

[101] Xiaoli Li and Zhongzhi Shi. Innovating web page classification through reducing noise. *J. Comput. Sci. Technol.*, 17(1):9–17, 2002.

[102] Shian-Hua Lin and Jan-Ming Ho. Discovering informative content blocks from web documents. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 588–593. ACM Press, Edmonton, Alberta, Canada, 2002.

[103] Jie Lu, Da Ruan, and Guangquan Zhang, editors. *E-Service Intelligence: Methodologies, Technologies and Applications*. Springer, 2007.

[104] Alexander Maedche. *Ontology learning for the semantic Web*. Kluwer international series in engineering and computer science ; SECS665. Kluwer Academic, Boston, 2002.

[105] Alexander Maedche, Viktor Pekar, and Steffen Staab. On discovering taxonomic relations from the web. In Ning Zhong, Jiming Liu, and Yiyu Yao, editors, *Web Intelligence*, chapter 14, pages 301–322. Springer, 2003.

[106] Alexander Maedche and Steffen Staab. Discovering conceptual relations from text. In *Proceedings of the 14th European conference on artificial intelligence*, pages 321–325. Berlin, Germany, 2000.

[107] Alexander Maedche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.

[108] F. Manola and E. Miller. Rdf primer. recommendation, w3c. http://www.w3.org/tr/2004/rec-rdf-primer-20040210, 2004.

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

[110] Deborah L. McGuinness. Ontologies come of age. In Dieter Fensel, J im Hendler, Henry Lieberman, and Wolfgang Wahlster, editors, *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential.* MIT Press, 2003.

[111] Deborah L. McGuinness and Frank van Harmelen. Owl web ontology language overview. w3c recommendation. http://www.w3.org/tr/2004/rec-owl-features-20040210, 2004.

[112] S.A. McIlraith, T.C. Son, and Honglei Zeng. Semantic web services. *Intelligent Systems, IEEE*, 16:46–53, 2001.

[113] Eduardo Mena, Arantza Illarramendi, Vipul Kashyap, and Amit P. Sheth. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases*, 8(2):223–271, 2000.

[114] M. Michalowski, S. Thakkar, and C. Knoblock. Exploiting secondary sources for automatic object consolidation. In *Proceedings of the 2003 ACM SIGKDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 34–36, Washington, DC, 2003.

[115] Marvin Minsky. A framework for representing knowledge. In P. Winston, editor, *The Psychology of Computer Vision.* McGraw-Hill, 1975.

[116] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, New York, 1997.

[117] Boris Motik, Ian Horrocks, Riccardo Rosati, and Ulrike Sattler. Can owl and logic programming live together happily ever after? In *Proceedings of the 5th International Semantic Web Conferent (ISWC 2006)*, 2006.

[118] I. Muslea, S. Minton, and CA Knoblock. Stalker: Learning extraction rules for semis-structured, web-based information sources. In *Proceedings of AAAI-98 Workshop on AI and Information Integration.* AAAI Press, 1998.

[119] Roberto Navigli, Paola Velardi, and Aldo Gangemi. Ontology learning and its application to automated terminology translation. *IEEE Intelligent Systems*, 18(1):22–31, 2003.

[120] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[121] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjorn Naeve, Mikael Nilsson, Matthias Palmer, and Tore Risch. Edutella: a p2p networking infrastructure based on rdf. In *Proceedings of the 11th international conference on World Wide Web*, pages 604–615. ACM Press, Honolulu, Hawaii, USA, 2002.

[122] Wolfgang Nejdl, Martin Wolpers, Wolf Siberski, Christoph Schmitz, Mario Schlosser, Ingo Brunkhorst, and Alexander Loser. Super-peer-based routing and clustering strategies for rdf-based peer-to-peer networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 536–543. ACM Press, Budapest, Hungary, 2003.

[123] H. B. Newcombe. *Handbook of Record Linkage: Methods for Health and Statistical Studies, Administration, and Business.* Oxford: Oxford University Press, 1988.

[124] H.B. Newcombe and J. M. Kennedy. Record linkage: Making maximum use of the discriminating power of identifying information. *Communications of the Association for Computing Machinery*, 5:563–567, 1962.

[125] H.B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.

[126] The U.S. Government Accountability Office. 9/11 commission report: Reorganization, transformation, and information sharing (gao-04-1033t). Technical report, The U.S. Government Accountability Office, 2004.

[127] Borys Omelayenko. Learning of ontologies for the web: the analysis of existent approaches. In *Proceedings of the International Workshop on Web Dynamics, held in conj. with the 8th International Conference on Database Theory (ICDT01)*, 2001.

[128] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP 2002*. 2002.

[129] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 251–260, 1995. TY - CONF.

[130] Eric Prudhommeaux and Andy Seaborne. Sparql query language for rdf (http://www.w3.org/tr/rdf-sparql-query/), 2007.

[131] Thanh Tho Quan, Siu Cheung Hui, A.C.M. Fong, and Tru Hoang Cao. Automatic generation of ontology for scholarly semantic web. In *Proceedings of the 3rd International Semantic Web Conference (ISWC 2004)*, pages 726–740. 2004. TY - BOOK.

[132] Dave Raggett, Arnaud Le Hors, and Ian Jacobs. Html 4.01 specification (http://www.w3.org/tr/html4/), 12 1999.

[133] Alan Rector and Jeremy Rogers. Ontological issues in using a description logic to represent medical concepts: Experience from galen. In *Proceedings of the IMIA WG6 Workshop: Terminology and Natural Language in Medicine*, 1999.

[134] Alan Rector, Jeremy Rogers, Angus Roberts, and Chris Wroe. Scale and context: Issues in ontologies to link health- and bio-informatics. In *Proceedings of the AMIA 2002 Annual Symposium*, pages 642–646, 2002.

[135] M. A. Roth, D. C. Wolfson, J. C. Kleewein, and C. J. Nelin. Information integration: A new generation of information technology. *IBM Systems Journal*, 41(4):563–577, 2002.

[136] Daniel L. Rubin, Micheal Hewett, Diane E. Oliver, Teri E. Klein, and Russ B. Altman. Automatic data acquisition into ontologies from pharmacogenetics relational data sources using declarative object definitions and xml. In *Proceedings of the Pacific Symposium on Biology*, 2002.

[137] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.

[138] Alexander Schutz and Paul Buitelaar. Relext: A tool for relation extraction from text in ontology extension. In *Proceedings of the 4th International Semantic Web Conference (ISWC 2005)*, 2005.

[139] N. Shadbolt, W. Hall, and T. Berners-Lee. The semantic web revisited. *Intelligent Systems, IEEE*, 21(3):96–101, Jan.-Feb. 2006.

[140] Parag Singla and Pedro Domingos. Collective object identification. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 1636–1637. Professional Book Center, 2005.

[141] Michael K. Smith, Chris Welty, and Deborah L. McGuinness. Owl web ontology language guide, 12 2003.

[142] Ruihua Song, Haifeng Liu, Ji-Rong Wen, and Wei-Ying Ma. Learning block importance models for web pages. In *Proceedings of the 13th international conference on World Wide Web (WWW '04)*, pages 203–211, New York, NY, USA, 2004. ACM Press.

[143] Adam Souzis. Building a semantic wiki. *IEEE Intelligent Systems*, 20(5):87–91, 2005.

[144] Peter Spyns, Robert Meersman, and Mustafa Jarrar. Data modelling versus ontology engineering. *SIGMOD Rec.*, 31(4):12–17, 2002.

[145] S. Staab and A. Maedche. Ontology engineering beyond the modeling of concepts and relations. In *Proceedings of the ECAI'2000 Workshop on Application of Ontologies and Problem-Solving Methods. IOS Press, Amsterdam.*, 2000.

[146] Ljiljana Stojanovic, Nenad Stojanovic, and Raphael Volz. Migrating data-intensive web sites into the semantic web. In *Proceedings of the 17th ACM symposium on applied computing (SAC)*, pages 1100–1107, 2002.

[147] H. Stuckenschmidt and Frank van Harmelen. *Information sharing on the semantic web.* Springer, Berlin, 2004.

[148] York Sure, Michael Erdmann, Juergen Angele, Steffen Staab, and Rudi Studer Dirk Wenke. Ontoedit: Collaborative ontology development for the semantic web. In *Proceedings of the 1st International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002*, 2002.

[149] Hendra Suryanto and Paul Compton. Discovery of ontologies from knowledge bases. In *Proceedings of the First International Conference on Knowledge Capture*, pages 171–178, 2001.

[150] Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning*. MIT Press, 2006.

[151] Roberto Tazzoli, Paolo Castagna, and Stefano Emilio Campanini. Towards a semantic wiki wiki web. In *Proceedings of the 3rd International Semantic Web Conference (ISWC2004)*, 2004.

[152] Sheila Tejada, Craig A. Knoblock, and Steven Minton. Learning object identification rules for information integration. *Information Systems Journal*, 26(8):635–656, 2001.

[153] Sheila Tejada, Craig A. Knoblock, and Steven Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '02)*, pages 350–359, New York, NY, USA, 2002. ACM Press.

[154] Jeffrey D. Ullman. Information integration using logical views. *Theor. Comput. Sci.*, 239(2):189–210, 2000.

[155] M. Uschold and M. Gruninger. Ontologies: principles, methods and applications. *The Knowledge Engineering Review*, 11(2):93–136, 1996. Publisher: Cambridge University Press, UK.

[156] Mike Uschold. Creating, integrating and maintaining local and global ontologies. In *Proceedings of the First Workshop on Ontology Learning in conjunction with the 14th European Conference on Artificial Intelligence*, 2001.

[157] Vladimir Naumovich Vapnik. *The nature of statistical learning theory*. Statistics for engineering and information science. Springer, New York, 2nd edition, 1999.

[158] Max Völkel, Markus Krötzsch, Denny Vrandecic, Heiko Haller, and Rudi Studer. Semantic wikipedia. In *Proceedings of the 15th international conference*

on World Wide Web (WWW '06), pages 585–594, New York, NY, USA, 2006. ACM Press.

[159] H. Wache, Th. Scholz, H. Stieghahn, and B. Konig-Ries. An integration method for the specification of rule-oriented mediators. In *Proceedings of the 1999 International Symposium on Database Applications in Non-Traditional Environments (DANTE '99)*, pages 109–112, Washington, DC, USA, 1999. IEEE Computer Society.

[160] H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neummann, and S. Huebner. Ontology-based integration of information-a survey of existing approaches. In *Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing*, pages 108–117. Seattle, WA, 2001.

[161] Chao Wang, Jie Lu, and Guangquan Zhang. A framework for capturing domain knowledge via the web. In *Proceedings of the eleventh Australasian World Wide Web Conference*, pages 248–255, 2005.

[162] Chao Wang, Jie Lu, and Guangquan Zhang. Mining key information of web pages. In *Proceedings of the 1st International Workshop on E-Service Intelligence in conjunction with 8th Joint Conference on Information Sciences.*, pages 1573–1576, 2005.

[163] Chao Wang, Jie Lu, and Guangquan Zhang. A semantic classification approach for online product reviews. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2005)*, pages 276–279, 2005.

[164] Chao Wang, Jie Lu, and Guangquan Zhang. Integration of ontology data through learning instance matching. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006)*, pages 536–539, 2006.

[165] Chao Wang, Jie Lu, and Guangquan Zhang. A constrained clustering approach to duplicate detection among relational data. In *Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2007)*, pages 308–319, 2007.

[166] Chao Wang, Jie Lu, and Guangquan Zhang. Generation and matching of ontology data for the semantic web in a peer-to-peer framework. In *Proceedings of the Joint Conference of the 9th Asia-Pacific Web Conference and the 8th International Conference on Web-Age Information Management (APWeb/WAIM 2007)*, pages 136–143, 2007.

[167] Chao Wang, Jie Lu, and Guangquan Zhang. Mining key information of web pages: a method and its application. *Expert Systems with Applications*, 33(2):425–433, 2007.

[168] Yalin Wang and Jianying Hu. A machine learning based approach for table detection on the web. In *Proceedings of the 11th international conference on World Wide Web (WWW '02)*, pages 242–250, New York, NY, USA, 2002. ACM Press.

[169] Sung-Shun Weng, Hsine-Jen Tsai, Shang-Chia Liu, and Cheng-Hsin Hsu. Ontology construction for information classification. *Expert Systems with Applications*, 31(1):1–12, 2006.

[170] Jennifer Widom. Research problems in data warehousing. In *Proceedings of the fourth international conference on Information and knowledge management (CIKM '95)*, pages 25–30, New York, NY, USA, 1995. ACM Press.

[171] Wikipedia. Peer-to-peer [http://en.wikipedia.org/wiki/P2p] — wikipedia, the free encyclopedia., 2006.

[172] William E. Winkler. Methods for record linkage and bayesian networks. Technical report, U.S. Census Bureau, Statistical Research Division, 2002.

[173] William E Winkler. Overview of record linkage and current research directions. Technical report, U.S. Census Bureau, Statistical Research Division, 2006.

[174] Y.Y. Yao, Ning Zhong, Jiming Liu, and Setsuo Ohsuga. Web intelligence (wi): Research challenges and trends in the new information age. In *Proceedings of the First Asia-Pacific Conference on Web Intelligence: Research and Development*, pages 1–17, 2001.

[175] Lan Yi and Bing Liu. Web page cleaning for web mining through feature weighting. In *Proceedings of Eighteenth International Joint Conference on Artificial Intelligence*, pages 43–50. 2003.

[176] Ning Zhong. Toward web intelligence. In *Advances in Web Intelligence: Proceedings of the First International Atlantic Web Intelligence Conference, Madrid, Spain, May 5-6, 2003*, pages 1–14, 2003.