Methods and systems for ontology learning, exploitation, and analysis

Xing, Jiang

2010

Xing, J. (2010). Methods and systems for ontology learning, exploitation, and analysis.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/40234>

<https://doi.org/10.32657/10356/40234>

Methods and Systems for Ontology Learning, Exploitation, and Analysis

Xing Jiang
School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2010

Abstract

While keyword based techniques continue to be the most popular option for information services, the limitations inherent in keywords routinely generate unsatisfactory results. As a promising alternative, ontology based solutions have been proposed to provide effective information services by exploiting ontologies for representing and organizing information. This thesis addresses the key issues in adopting ontology based solutions by presenting a collection of methods and systems for ontology building, ontology exploitation, and ontology analysis.

In any ontology based solution, ontologies firstly have to be created for representing and organizing information. However, ontology building is well known to be a tedious process. Manually acquiring knowledge for building domain ontologies requires much time and resources. To ease the efforts of building ontologies, we develop a system called Concept-Relation-Concept Tuple based Ontology Learning (CRCTOL) for **automatically learning ontologies from domain specific text documents**. By using a full text parsing technique and incorporating both statistical and lexico-syntactic methods, the ontologies learned by our system are more concise and contain a richer semantics in terms of the range and number of semantic relations compared with alternative systems.

To provide ontology assisted services, we study a major application of ontology based solutions, namely **ontology based information retrieval**. In view of the limitations of the existing user models, we develop an ontology based user model, called *user ontology*, which is a specialization of the domain ontology by assigning each concept and relation of the domain ontology with a specific value for indicating a user's interests. We have developed methods for learning and inferencing in the user ontology model and integrated it into a semantic search engine called *OntoSearch* for providing personalized document retrieval. The experimental results, based on the ACM digital library and the Google

Directory, support the efficacy of the user ontology model and the validity of learning and exploiting user ontology.

For enabling up-to-date services, ontologies have to evolve over time. To support a continual ontology evolution, we apply frequent subgraph mining techniques for **ontology instance data driven change discovery**. To this end, a specific type of graph mining problems, i.e., *Mining Globally Distributed Frequent Subgraphs in a Single Labeled Graph*, is identified. The graph patterns discovered, called *G-Patterns*, can provide global and balanced information about the entire graph and are valuable for a variety of applications. For mining G-Patterns, an objective measure called *G-Measure* and a *G-Miner* algorithm are developed. The experimental results on both synthetic and real-world data show that the G-Miner algorithm can effectively discover G-Patterns and run much faster than the alternative algorithms.

Acknowledgments

This thesis is the result of my work as a Ph.D. student at the **School of Computer Engineering, Nanyang Technological University**. Many people have supported me and made it possible for completing this thesis successfully. Here, I would like to take this opportunity to express my thanks and gratitude to them.

First of all, I would like to express my sincere thanks to my supervisor, **Professor Ah-Hwee Tan**, for his years of encouragement and support during my research. He is a wonderful supervisor who allows me an ideal level of freedom in pursuing my ideas, while at the same time consistently offers advices to keep me focused. Without his guidance, the thesis would not have been completed.

I would like to thank **Professor Hui Xiong** for sharing his ideas and providing critical, but extremely useful comments for the draft of the papers and thesis. I would also like to thank **Dr. Chen Wang** who offers me the internship in the IBM China Research Laboratory and lets me think about new angels of my research.

The community of faculties and students in the school has provided me an excellent work environment. I would like to thank them all. Some deserve special mention for this thesis: **Ms. Huan Wang, Dr. Tao Jiang, Dr. Jianshu Weng, Professor Aixin Sun**, and **Professor Clement Liang-Tien Chia**.

Finally, and most importantly, thanks to my parents for their unwavering love and support during and before this work. The thesis is dedicated to them.

Thank you!

Contents

Abstract	i
Acknowledgments	iii
List of Figures	viii
List of Tables	xii
1 Introduction	1
1.1 Background	1
1.2 General Approaches	3
1.3 Ontology Based Solution	6
1.3.1 Ontology Building	6
1.3.2 Ontology Exploitation	8
1.3.3 Ontology Analysis	10
1.4 Summary of Contributions	11
1.5 Thesis Structure	13
2 Related Work	14
2.1 What is an Ontology?	14
2.2 Ontology Learning	15
2.2.1 Ontology Learning from Domain Specific Text Documents	15
2.2.2 Ontology Learning from Database Schemas	17
2.2.3 Ontology Learning from Machine Readable Dictionary	19
2.2.4 Ontology Learning from Wikipedia	20
2.3 Ontology Based Information Retrieval	22
2.3.1 Searching With Ontological Annotations	22
2.3.2 Searching With Keywords	29

2.4	Ontology Based User Modelling	30
2.5	Ontology Analysis	33
2.5.1	Ontology Driven Change Discovery	34
2.5.2	Usage Driven Change Discovery	35
2.5.3	Data Driven Change Discovery	36
2.6	Summary	38
3	Mining Ontological Knowledge from Domain Specific Text Documents	39
3.1	Introduction	39
3.2	System Architecture	40
3.3	Concept Extraction	44
3.3.1	Concept Extraction Procedure	44
3.3.2	Concept Extraction Measure	47
3.4	Word Sense Disambiguation	51
3.4.1	LESK Algorithm	52
3.4.2	WordNet	52
3.4.3	VLESK Algorithm	53
3.5	Semantic Relation Extraction	55
3.5.1	Taxonomic Relation Extraction	55
3.5.2	Non-taxonomic Relation Extraction	57
3.6	Ontology Assembling	59
3.6.1	Taxonomic Relation Assembling	59
3.6.2	Non-taxonomic Relation Assembling	60
3.7	Experiments	60
3.7.1	Component Level Evaluation	62
3.7.2	Ontology Level Evaluation	71
3.8	Summary	80
4	Learning and Inferencing in User Ontology for Personalized Information Services	82
4.1	Introduction	82
4.2	User Ontology Model	84

4.3	Spreading Activation Theory	86
4.4	User Ontology Learning	88
4.4.1	Learning Concepts of Interest	88
4.4.2	Learning Relations of Interest	89
4.4.3	An Illustration	89
4.5	OntoSearch: A Full-Text Search Engine	91
4.5.1	Motivation	91
4.5.2	Approach and Assumption	92
4.5.3	System Flow	93
4.5.4	Ontological Indexing	95
4.5.5	Inferencing in User Ontology	95
4.5.6	Similarity Measures	96
4.6	Experiments	98
4.6.1	Searching ACM Digital Library	98
4.6.2	Searching Google Directory	103
4.6.3	Discussion	106
4.7	Summary	109
5	Mining Ontology Instance Data For Change Discovery	111
5.1	Introduction	111
5.2	Preliminary Concepts	114
5.3	G-Measure	118
5.3.1	Background	118
5.3.2	Basic Idea	119
5.3.3	Definition	120
5.3.4	Conditions of Downward Closure Property	122
5.3.5	An Approximate Method of Computing G-Measure	128
5.4	G-Miner	131
5.4.1	The DFS code	132
5.4.2	Algorithm Details	133
5.5	Experiments	136

5.5.1	Experimental Setup	136
5.5.2	Effectiveness for Mining G-Patterns	141
5.5.3	Computational Efficiency of Mining G-Patterns	144
5.6	Analysis of China's Stock Market: A Case Study	145
5.7	Related Work	149
5.8	Summary	151
6	Conclusions and Future Work	152
6.1	Contributions	152
6.1.1	Ontology Building	152
6.1.2	Ontology Exploitation	153
6.1.3	Ontology Analysis	154
6.2	Outstanding Issues	155
6.2.1	Ontology Building	155
6.2.2	Ontology Exploitation	156
6.2.3	Ontology Analysis	157
	Appendix	158
A	List of Publications	158
A.1	Journals	158
A.2	Conferences	158
	References	160

List of Figures

1.1	Different keywords are used for describing a same concept.	2
1.2	The statistics of the monthly tweets published in Singapore since 2006. .	3
1.3	An example of web pages annotated with concepts for specifying their meanings.	5
1.4	The life cycle of an ontology based solution.	6
1.5	A partial sport event domain ontology.	7
2.1	The approach developed in the InfoSleuth project for ontology learning from database schemas.	18
2.2	An example to illustrate how SearchMonkey works.	23
2.3	A screen shot of Goat's interface.	25
2.4	The SPARQL query for retrieving all country capitals in Asia.	27
2.5	A screen shot of the SHOE search engine.	28
2.6	An example to illustrate the user model built in [PG99], where the values assigned to each concept represent the user's degree of interest to this concept.	31
2.7	An illustration of the distance based inference method.	32
2.8	The two different ways of defining a class in RDF schema.	35
2.9	An example of the ontology instance data.	37
3.1	The architecture of the CRCTOL system.	41
3.2	A sample input sentence and the corresponding output of the NLP component.	42
3.3	The user interface.	43

3.4	An example to illustrate the concept extraction procedure, where the multi-word concept <i>terrorist group</i> and single-word concept <i>group</i> are extracted from texts.	46
3.5	A sample input and the corresponding output of the word sense disambiguation module.	51
3.6	A sample input and the corresponding output of the semantic relation extraction module.	55
3.7	The POS tags assigned for the sample sentence.	59
3.8	The parse tree of the sample sentence.	59
3.9	The Pseudo-Code of the non-taxonomic relation integration algorithm. .	60
3.10	The concept <i>militant group</i> extracted and the relations around it.	72
3.11	The degree distribution $p(k)$ and its Log-Log plot for the domain ontology built on the PGT data set.	74
3.12	The performance of Text-To-Onto, Text2Onto, and CRCTOL for concept extraction with different K values.	77
3.13	The relation <i>inMatch</i> defined in the benchmark ontology.	79
4.1	A partial domain ontology for the Italian soccer teams.	85
4.2	An illustration of the user ontology.	86
4.3	Initial stage of the spreading activation process.	87
4.4	Final stage of the spreading activation process.	87
4.5	The procedure of the OntoSearch system for enhanced document retrieval.	93
4.6	The Pseudo-Code of the OntoSearch system's overall algorithm.	94
4.7	An illustration of a document and its annotations in the ACM digital library data set.	99
4.8	The average precision of OntoSearch_U compared with Lucene, OntoSearch, and OntoSearch_C based on the top five documents retrieved.	101
4.9	The average precision of OntoSearch_U compared with Lucene, OntoSearch, and OntoSearch_C based on the top ten documents retrieved.	101
4.10	An illustration of the ODP taxonomy's structure, where each level represents a particular category.	103

4.11	The average precision of OntoSearch_U compared with Google Directory, OntoSearch, and OntoSearch_C based on the top five documents retrieved.	105
4.12	The average precision of OntoSearch_U compared with Google Directory, OntoSearch, and OntoSearch_C based on the top ten documents retrieved.	105
4.13	The average precision of OntoSearch_U on the top ten documents retrieved from the ACM digital library data set with different decay values for inferencing in the user ontology.	108
5.1	An example of the ontology instance data, where web pages are connected with hyperlinks and annotated with relevant concepts.	112
5.2	A labeled graph corresponding to the example ontology instance data, where the labels are the concepts used for annotating the web pages. . .	112
5.3	The frequent graph patterns found in Figure 5.2.	113
5.4	The illustration of a single labeled graph G , its subgraph SG , and a graph pattern P identified in G	114
5.5	An illustration of a graph pattern P and its four instances in the labeled graph G shown in Figure 5.4(a).	116
5.6	The edge-disjoint and vertex-disjoint based instance graphs of P	116
5.7	An example showing that the downward closure property does not hold for mining a single labeled graph.	117
5.8	The graph pattern P and the input graph G	120
5.9	The built P 's instance graph.	120
5.10	An example of the forming clique operation, where the vertices are highlighted with black color.	123
5.11	The sample patterns and the input graph corresponding to the forming clique operation shown in Figure 5.10.	123
5.12	An example of the removing vertex operation, where the vertex is highlighted with black color.	124
5.13	An example of the adding edge operation, where the vertices are highlighted with black color.	124
5.14	The sample patterns and the input graph corresponding to the adding edge operation shown in Figure 5.13.	125

5.15	The sample patterns and the input graph for illustrating how the instance graph of Q is constructed from that of P with the three operations. . . .	125
5.16	The detailed steps of building Q 's instance graph.	126
5.17	Deleting an edge from P 's instance graph for forming Q 's instance graph.	127
5.18	Adding a new vertex into Q 's instance graph whose corresponding instance is not generated from P 's instance.	127
5.19	An example showing that the G-Measure does not have the downward closure property.	128
5.20	The two approaches of computing G-Measure value.	129
5.21	An illustration of the DFS code based approach.	132
5.22	The Pseudo-Code of the G-Miner Algorithm.	134
5.23	A 4-edge pattern with frequency of 1,843,068 in the original citation graph.	138
5.24	The pseudo-code of the baseline-1 algorithm.	139
5.25	The pseudo-code of the baseline-2 algorithm.	140
5.26	The coverage of the top ten frequent multi-edge patterns extracted by G-Miner compared with that of baseline-1 on graph V2E1N500L20A55. .	142
5.27	The coverage of the top ten frequent multi-edge patterns extracted by G-Miner compared with that of baseline-1 on graph V2E1N500L20A05. .	142
5.28	The coverage of the top ten frequent multi-edge patterns extracted by G-Miner compared with that of baseline-1 on graph V2E1N550L10A65. .	143
5.29	The coverage of the top ten frequent multi-edge patterns extracted by G-Miner compared with that of baseline-1 on graph V2E1N550L100A65. .	143
5.30	An example of the CSM data set.	146
5.31	The number of G-Patterns mined from the graphs CSM2002 and CSM2005 with different minimum support value and the runtime.	146
5.32	Sample patterns mined from the graphs CSM2002 with minsup 100. . .	147
5.33	Sample patterns mined from the graphs CSM2005 with minsup 100. . .	148

List of Tables

2.1	The concept frame of the concept “computer virus”.	16
3.1	Contingency table.	50
3.2	The senses of cone and pine in the dictionary.	52
3.3	The 18 sense combinations of “ <i>international (ADJ) terrorist(ADJ) at- tack(NOUN)</i> ”.	54
3.4	The lexico-syntactic patterns used for taxonomic relation extraction. . . .	56
3.5	The performance of Text-To-Onto, Text2Onto, and CRCTOL in multi- word term extraction.	63
3.6	The performance of the re-implemented POS tag based rule for extracting multi-word terms from the texts processed by the Berkeley parser and the Stanford parser.	63
3.7	The performance of Text-To-Onto, Text2Onto and CRCTOL in concept extraction.	64
3.8	The performance of the re-implemented Text-To-Onto and Text2Onto for extracting concepts from the texts processed by the Berkeley parser and the Stanford Parser.	65
3.9	The performance of CRCTOL compared with those of the re-implemented Text-To-Onto and Text2Onto for concept extraction on the new test corpus. .	66
3.10	The performance of the VLESK algorithm compared with the two baseline algorithms.	68
3.11	The performance of Text-To-Onto, Text2Onto and CRCTOL for non- taxonomic relation extraction on simple structure sentences.	69
3.12	The performance of the POS tag based rules for non-taxonomic relation extraction on simple structure sentences.	69

3.13	The performance of Text-To-Onto, Text2Onto, and CRCTOL for non-taxonomic relation extraction on general sentences.	70
3.14	The performance of the POS tag based rules for non-taxonomic relation extraction on general sentences.	70
3.15	The top ten multi-word terms extracted.	72
3.16	Two dimensions of evaluating the quality of the built domain ontology, rated by five human judges.	75
4.1	The average precision and standard deviation of Lucene, OntoSearch, OntoSearch_C and OntoSearch_U on the top five and ten documents retrieved.	102
4.2	The p -values for the paired t-tests on the ACM digital library data set. . .	103
4.3	The average precision and standard deviation of Google Directory, OntoSearch, OntoSearch_C and OntoSearch_U on the top five and ten documents retrieved.	107
4.4	The p -values for the paired t-tests on the Google Directory data set. . . .	107
4.5	Analysis of the performance gain brought by relation learning on the ACM digital library data set and the Google Directory data set.	108
4.6	The average precision and standard deviation of Onto_U on the top ten documents retrieved from the ACM digital library data set with different decay values for inferencing in the user ontology.	109
5.1	Parameter settings for the synthetic data used in the experiments.	136
5.2	The characteristics of the real-world data.	137
5.3	The number of unique vertices in the patterns found in the synthetic data.	144
5.4	The experimental results of G-Miner, baseline-1, and baseline-2 on the six real-world data.	145
6.1	Summary of key features compared with other ontology learning systems.	153

Chapter 1

Introduction

1.1 Background

“Ron, a freshman of the university, wants to buy a car so that he does not spend more than an hour every day to wait for the shuttle buses. Before making the purchase, Ron would like to collect as much useful information as possible with Google. However, after submitting the query to Google, Ron finds the task astonishing as there are more than one million documents containing the keyword *car*, many of which he may have to look at. In the meantime, as some agents and web sites use *automobile* instead of *car* in the documents, Ron may need to search again with the keyword *automobile* and browse another one million documents retrieved (see Figure 1.1).”

“Jessica, a geneticist in the national lab, needs to search for some genes from the Entrez Gene database (<http://www.ncbi.nlm.nih.gov/sites/entrez>). Simply typing the keyword *piwi*, there are a total of 726 records returned, as it is unlucky that *piwi* turns out to be used as the name, symbol, or alias of a number of different genes belonging to human, mice, and rats. Unfortunately, after filtering out the irrelevant species, there are still 16 genes of homo sapiens left, some of which (e.g., *hiwi*) are associated with proliferation of cancer cells, some of which (e.g., *hiwil2*) are related to germline and stem cell, and some of which the functions have not been identified. Jessica has to check all the 16 genes to avoid taking wrong samples for her experiments.”

CHAPTER 1. INTRODUCTION

Figure 1.1: Different keywords are used for describing a same concept.

The above scenarios are just a few of the problems we are now facing in our daily life. On the one side, with the rapid development of computer technologies, much more information is being produced. For example, Figure 1.2 presents the statistics of the monthly tweets published in Singapore since 2006¹. We can see many more tweets have been published recently. Also, Google recently announces that it has indexed one trillion unique URLs.² There is certainly something useful that has been made. On the other side, the traditional keyword based approach is still adopted to represent and organize information. It is simple but ineffective and meaningless, especially for software agents, making it even harder to satisfy people's requirements with respect to the huge amount of information being produced. The synonyms and ambiguity problems, just as shown in the above scenarios, can easily frustrate people when searching for a particular item, integrating different data resources, or sharing information. More importantly, no ad-

¹The statistics are obtained with twitter API <http://apiwiki.twitter.com/>

²<http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html>, retrieved on 21/May/2009.

Figure 1.2: The statistics of the monthly tweets published in Singapore since 2006.

vanced functions are supported by the keyword based approach. For example, since the launch of the first Web search engine ALIWEB in 1993, we can see the current search engines, e.g., Google and Yahoo, still follow such a paradigm of search. People have to specify their requirements carefully in a set of keywords and hope that the top document retrieved by the search engine is exactly the required one. There are no better ways for document retrieval. As a result, people cannot fully enjoy the benefits brought when a huge amount of information is available.

1.2 General Approaches

In view of the problems, two families of solutions are proposed to provide information services in totally opposed ways.

Straightforwardly, software agents which are as intelligent as human being are expected to be invented. They can easily understand the content of a document or correctly capture the semantics in a picture when fulfilling people's requirements. However, despite the latest advances in many disciplines, it remains hard to develop such agents within the

CHAPTER 1. INTRODUCTION

current condition. Taking the winner of the TREC Query Answering Track 2007 as an example [DKL07], it still has serious troubles in answering questions like *What were the names of characters in the TMNT stories* and *What is the name of the Wizarding Game in “Harry Potter and the Goblet of Fire”*, which can be easily answered by a preschool child. There is quite a long way to reach this goal.

Alternatively, ontology based solutions are proposed that utilize ontologies to represent and organize information for providing services. In philosophy, *Ontology* is a part of *metaphysics*, which is the study of nature of being, existence or reality in general. In computer science, an *ontology* is a formal, explicit specification of a shared conceptualisation [Gru93]. It may be as simple as the Yahoo Directory³ that contains only a set of concepts and taxonomic relations between concepts or as comprehensive as the Cyc ontology [GL90] which is full of concepts, relations between the concepts, and axioms on the relations in everyday common sense knowledge. But in general, an ontology defines the standard terms to represent an area of knowledge⁴. By accessing the terms specified in the ontologies instead of the keywords, the effective communication among people and software agents becomes possible. For example, if the web pages are annotated with the particular concepts of the ontology which they are related to (an example is given in Figure 1.3), Ron just needs to use the concept *car* to search documents. Also, the database is able to provide Jessica the precise information, since the difference between *piwi* (*sensu Homo sapiens cancer*) and *piwi* (*sensu Mus musculus germline*) has been specified.

Meanwhile, an ontology makes the domain knowledge and human assumptions explicit by presenting the relationships between concepts and the axioms on relations in a clear way. The software agents can thus perform human-like reasoning and further conduct complicated tasks for the users. For example, when looking for unique foxes in Africa,

³<http://dir.yahoo.com/>

⁴<http://www.w3.org/TR/webont-req/>, retrieved on 23/May/2009.

CHAPTER 1. INTRODUCTION

Figure 1.3: An example of web pages annotated with concepts for specifying their meanings.

as the ontology has specified that bat-eared fox lives only in South Zambia, Botswana, Namibia, and South Africa and the four countries are all in Africa, the search engine can still get the correct result even though no documents that indicate bat-eared fox is in Africa exist. No advanced techniques but a few of inference rules are employed in this search process. The provision of high quality services becomes possible.

Besides the above two points, the use of ontologies for representing and organizing information will also offer many other possibilities such as enabling reuse of domain knowledge and separating domain knowledge from operational knowledge. Its advantages are obvious, while the realization is much more practical compared with the former approach of developing intelligent software agents. In fact, many important technologies such as the eXtensible Markup Language (XML)⁵ and Resource Description Framework (RDF)⁶ for supporting the use of ontologies are already in place. Consequently, ontology based solutions are quickly adopted in many disciplines such as the Semantic Web [BLHL01], bioinformatics [ABB⁺00, SK02], software engineering [CFM06], and information system [Gua98].

⁵<http://www.w3.org/XML/>

⁶<http://www.w3.org/RDF/>

Figure 1.4: The life cycle of an ontology based solution.

1.3 Ontology Based Solution

Given the different roles an ontology plays in the process of providing services, we can divide the life cycle of an ontology based solution into three stages (see Figure 1.4), namely ontology building, ontology exploitation, and ontology analysis. Ontology building concerns the construction of an ontology for the domain of interest. Its output is then exploited in different kinds of services. Finally, the built ontologies, the system's performance, and the feedback are analyzed, which in turn assists the construction of a new ontology or the refinement of an existing ontology. This cycle repeats, resulting in the proliferation of high quality ontologies and the success of the ontology based solution.

1.3.1 Ontology Building

In general, an ontology consists of concepts and relations between concepts in the domain of interest. An example is shown in Figure 1.5, which presents a partial sport event domain ontology. Traditionally, **ontology engineering**, a field in knowledge representation and artificial intelligence, studies the ways of building such a shared understanding [GPCFL04]. Guided by the proposed ontology engineering methodologies

CHAPTER 1. INTRODUCTION

Figure 1.5: A partial sport event domain ontology.

(e.g., [LG89, UK95, GF95, BLC96, SRKR97, LGPSS99, SSSS01]) and assisted with a set of tools (e.g., Ontolingua Server [FFR97], Prott-2000 [NFM00], OntoEdit [SEA⁺02]), people have manually built ontologies, such as Sowa’s top level ontology [Sow99], Cyc’s upper ontology [LG89], and Suggested Upper Merged Ontology (SUMO) [PNL02].

However, this manual construction process is extremely costly and time-consuming, as an ontology is supposed to have a significant coverage of the domain of interest. Taking WordNet [Fel98] as an example, it needs the human editors to manually input about 150,000 words and over 115,000 synsets for a total of 207,016 word-sense pairs. The construction of WordNet started in 1985 and is still not finished yet. Automatic techniques are strongly needed for building ontologies.

Furthermore, the construction of a comprehensive ontology covering everything is complicated and in fact impossible, as the ontology has to be agreed by different parties in a community and divergence is unavoidable. Simply requiring everyone to share

exactly the same definition of concepts and relations as in traditional knowledge representation systems, e.g., [BS84], can only result in people's resistance to the ontology based solution. In contrast, the construction of a series of small size ontologies for covering the whole universe is more practical [BLHL01]. The trade-off between the coverage and the conciseness of the ontology makes the construction process indeed a challenging problem, which has been referred to as the *knowledge acquisition bottleneck* [UK95, LGPSS99].

In view of the above problems, **ontology learning** [MS01] is proposed to automatically or semi-automatically learn ontologies from domain relevant resources. For example, because concepts are normally nouns in texts, we can quickly find a set of relevant concepts by extracting nouns from the domain specific documents. Compared with the manual approach, the costs are greatly reduced. As there has been a huge amount of data available and we do not need a comprehensive ontology covering everything, ontology learning becomes an ideal solution for building ontologies and effective ontology learning methods are being explored [GPMM].

1.3.2 Ontology Exploitation

Many applications have been developed to exploit the possible ways of using ontologies for providing services such as document retrieval and image retrieval. According to the different ways how an ontology is used, we could classify the existing applications into two categories.

- (i) **Ontology Based Specification:** The use of ontologies in the first category of applications is mainly because an ontology can provide a controlled and uniform vocabulary to specify the domain of interest. For example, as each document is annotated with specific terms in the ontology specifying its meaning, OntoSeek [GMV99] could retrieve the precise product information, e.g., car but not motor,

for satisfying a user's requirement. Also, after the clients' requirements are characterized and specified using terms in the ontology, the validation and verification of software can be easily conducted [UG04]. In addition, the use of ontologies for content management enables the reuse of knowledge, increasing maintainability and long term knowledge retention in an organization [MISA04]. This is the simplest way of using an ontology. However, the benefit brought is already significant and most applications adopt this approach.

- (ii) **Ontology Based Reasoning:** Ontology based specification is not the only way of using ontologies. Note that an ontology not only includes standard terms for representing the domain of interest but also encodes the domain knowledge and human assumptions in an explicit way by a set of relations and axioms. The effective use of the explicit domain knowledge and human assumptions by the suitable ontology based reasoning methods such as *subsumption*, *consistent checking*, and *rule-based reasoning* can further improve the quality of the services provided. For example, when browsing web pages of a particular concept, the relevant documents are recommended as their associated concepts are linked to the current interest with certain relations but not by simply analyzing a set of historical data. It makes the recommendation conducted in a more human-like manner [HMS⁺05]. Furthermore, as new or implicit knowledge can be derived, the performing of complex tasks becomes possible. For example, given a patient's CT image, the doctor can quickly determine which organs are injured given the particular trajectories of projectiles, whether vital structures such as a coronary artery are injured, and predict the propagation of injury ensuing after vital structures are injured with the help of ontologies [RDB⁺06]. This is a promising way of using ontologies and being exploited by many applications.

1.3.3 Ontology Analysis

Ontologies conceptualize the domain of interest and are the backbone of ontology based applications. However, ontologies are not static. Domain changes, adaptations to different tasks, or changes in the conceptualization all require the modification of the existing ontology or the construction of a new ontology. If the underlying ontology is not up-to-date, the reliability, accuracy and effectiveness of the system all would decrease significantly [KF01]. Ontology analysis is aimed at capturing the necessary changes in the ontologies for facilitating a continual evolution of ontologies.

In the early stage, ontology analysis is directly operating on the ontologies to capture the changes, where the ontology engineers play a major role. For example, by distinguishing the different versions of ontologies and keeping track of their relationships, the modifications performed by the human experts are verified and the compatibility between the different versions is made explicit. The suitable parts of the existing ontologies can thus be reused in new situations without invalidating the current usage [KF01].

Later on, the end users' responses when interacting with the ontologies are utilized for ontology analysis. The typical usage patterns found from the end users' behaviors reflect the users' interests in part of the ontologies, which illustrates the anomalies in the ontologies and gives concrete clues on how the ontology should be improved [SSGS03].

The above two approaches have shown their effectiveness in supporting the evolution of ontologies. However, they both require extensive human involvement. Given the fact that ontologies are able to be learned from suitable data, **data driven change discovery** is proposed that derives ontology changes from suitable data as well [Sto04]. For example, if no instances of a concept C can be found in the ontology instance data, it indicates that C is not important and could be removed from the ontology. As most parts are done automatically, data driven change discovery would greatly facilitate the evolution of ontologies. Consequently, it becomes a promising approach of ontology analysis, where

techniques such as Formal Concept Analysis (FCA), or heuristics based methods have been employed.

1.4 Summary of Contributions

In this thesis, we address the issues of adopting ontology based solutions for providing services, using information retrieval as the specific application domain. In particular, new techniques and algorithms are developed for ontology learning from domain specific text documents, ontology based user modelling, ontology based information retrieval, and ontology instance data driven change discovery. The main contributions are summarized as follows:

- (i) **Ontology Learning:** Domain specific text documents provide a direct resource for learning ontologies. However, traditional ontology learning systems employ shallow natural language processing techniques and focus only on concept and taxonomic relation extraction. As a result, the learned ontologies have simple semantics and may not have a high quality. In view of the problems, we develop a system, called Concept-Relation-Concept Tuple based Ontology Learning (CRCTOL), for mining ontologies automatically from domain specific text documents. Specifically, CRCTOL adopts a full text parsing technique and employs a combination of statistical and lexico-syntactic methods, including a statistical algorithm that extracts key concepts from a document collection, a word sense disambiguation algorithm that disambiguates words in the key concepts, a rule based algorithm that extracts relations between the key concepts, and a modified generalized association rule mining algorithm that prunes unimportant relations, for ontology learning. As shown in the experimental results, the ontologies learned by CRCTOL are more concise and contain a richer semantics compared with alternative systems. The work has been reported in [JT05, JT10].

- (ii) **Ontology Exploitation:** One major application of ontology based solutions is ontology based information retrieval, wherein many possibilities have been offered. In this thesis, we exploit the use of ontologies for performing personalized information retrieval. In particular, in view of the limitations of existing user models [MSR04, PG99, VCF⁺07, CNPK05], we develop an ontology based user model, called *user ontology*, for capturing a user's interests. As a customized view of the domain ontology, a user ontology can provide a richer and more precise representation of the user's interests in a domain of interest compared with the existing user models. We develop a set of statistical methods for learning individual user ontologies from an existing domain ontology and a spreading-activation theory (SAT) [And83b] based procedure for inferencing in the user ontology. The proposed user ontology model and the spreading-activation based inferencing procedure have been incorporated into a semantic search engine called *OntoSearch* which originally utilizes ontology with keywords for document retrieval and later is extended for image retrieval. The experimental results, based on the ACM digital library and the Google Directory, support the efficacy of the user ontology model and the validity of learning and exploiting user ontology. The work have been published in [JT06a, JT06b, WJCT08, JT09].
- (iii) **Ontology Analysis:** Given the availability of possible resources for ontology analysis, we adopt data driven change discovery for tackling this problem. Specifically, we encode the ontology instance data as a single labeled graph and apply frequent subgraph mining techniques for change discovery. To this end, a specific type of graph mining problems, i.e., *mining globally distributed frequent subgraphs in a single labeled graph*, is identified. The patterns introduced by this problem, called *G-Patterns*, can provide global and balanced information about the entire graph and are valuable for a variety of applications. For mining G-Patterns, an

objective measure called *G-Measure* and a G-Miner algorithm are developed. The experimental results on both synthetic and real-world data show that the G-Miner algorithm can effectively discover G-Patterns and run much faster than the alternative algorithms. This work can be found in [JXWT09].

1.5 Thesis Structure

The outline of this thesis is given as follows. We first review the existing approaches to ontology learning, ontology exploitation, and ontology analysis in Chapter 2. From Chapter 3 to Chapter 5, the new techniques and algorithms developed in this dissertation research are presented in details.

Chapter 3 presents the CRCTOL system which is aimed at automatically learning ontologies from domain specific text documents. Extensive experiments have been conducted for evaluating this system's performance.

Chapter 4 introduces methods for learning and inferencing in the user ontology model. In addition, the OntoSearch system is presented which integrates the user ontology model for performing personalized document retrieval.

Chapter 5 focuses on the ontology instance data driven change discovery. In particular, a new type of graph mining problems, i.e., *mining globally distributed frequent subgraphs in a single labeled graph*, is identified. Then, the G-Measure and the *G-Miner* algorithm are proposed to address this problem.

The final chapter recaps the main contributions of this thesis and discusses how the work could be taken further.

Chapter 2

Related Work

In this chapter, we review the state of the art in the research fields related to ontology based solutions. In particular, we first present the definition of ontology which we use in this thesis. Then, we focus on four related research areas, namely ontology learning, ontology based information retrieval, ontology based user modeling, and ontology analysis.

2.1 What is an Ontology?

The term *ontology* is borrowed from Philosophy, in which it means a systematic explanation of what entities can be said to exist, and how such entities can be related. In this thesis, we adopt the most quoted definition of *ontology* in the context of computer science, which is defined as follows [Gru93]:

An ontology is an explicit specification of a conceptualization.

Here, a conceptualization is an abstract, simplified view of a domain of interest which consists of the objects, concepts, and other entities that are presumed to exist in this domain of interest and the relationships that hold them [GN87]. Therefore, an ontology should contain

- The **concepts** in the domain of interest

- The **relations** that can exist among the concepts

Meanwhile, it requires that an ontology must be represented in an explicit specification. To fulfill this requirement, a representational, controlled **vocabulary** has to be defined for describing these concepts and relations in a formal way. By using the standard terms, the share of a conceptualization among a community of people and software agents becomes possible.

2.2 Ontology Learning

Research in ontology learning explores possible ways of building an ontology from scratch, or adapting an existing ontology in an *automatic* or *semi-automatic* fashion from relevant resources so as to avoid the costs of building an ontology manually [GPMM]. Currently, there are four types of important resources used for ontology learning, namely domain specific text documents, database schema, machine readable dictionary (MDR), and Wikipedia¹.

2.2.1 Ontology Learning from Domain Specific Text Documents

In this section, we review the systems and methods for ontology learning from domain specific text documents. Here, *domain specific text documents* refer to all textual data written in natural languages and not limited to files with *.txt* format. Compared with the other three resources, domain specific text documents tend to be widely available and publicly accessible, which makes them the major resource for ontology learning.

CORPORA-Ontobuilder [Eng03] is a system developed by CognIT in the ontology Knowledge project for learning ontologies (mainly containing taxonomic relations) from document specific text documents. In particular, Ontobuilder uses OntoWrapper to extract information from on-line resources (e.g. names, emails, addresses, etc.) and

¹<http://www.wikipedia.org/>

CHAPTER 2. RELATED WORK

Table 2.1: The concept frame of the concept “computer virus”.

Name: Computer Virus

Synset: Virus, Worm, Trojan Horse

Rels: (Computer Virus, infect, Computer), (Hacker, create, Computer Virus),
(Compute Virus, spread by, Email).

OntoExtract to obtain taxonomic relations from text. However, the details of how this system works are not released.

OntoLearn [MNV02] is an ontology learning system developed at IASI-CNR which makes use of shallow natural language processing (NLP) tools, including a morphologic analyzer, a part-of-speech (POS) tagger and a chunk parser, to process documents and employs text mining techniques to produce ontologies based on document collections. However, the performance of the concept extraction method is greatly affected by the size of the document collections used for ontology learning.

Text-To-Onto [MS00], also based on shallow NLP tools, is able to extract key concepts and semantic relations (including non-taxonomic ones) from texts. The selection of concepts is based on the tf/idf [SM86] measure used in the field of information retrieval. Semantic relations between concepts are extracted using an association rule mining algorithm and predefined regular expressions. However, as tf/idf is designed primarily for document retrieval but not for concept extraction, the system cannot effectively extract domain specific concepts. Also, the identification of semantic relations is based on POS tags only, limiting the accuracy of the relations extracted. The same problems are also suffered by its successor called Text2onto [CV05].

Rajaraman and Tan [RT03] extract knowledge in the form of concept frame graphs (CFGs) from text documents (an example of one extracted concept frame is given in Table 2.1, which is a node of the CFG). Semantic relations between concepts are extracted through analyzing the part-of-speech tags of the sentences using a library of extraction

rules. However, as the CFG system extracts concepts and relations from all sentences without considering their importance, it tends to extract a large number of concepts and relations, many of which have no real significance. Also, the CFG system is designed to extract non-taxonomic relations only. As a result, the system fails to recognize the taxonomic relations between concepts, for example, the taxonomic relation between *Computer Virus* and *Virus* in the above example.

Nováček *et al.* from DERI [NLHD08] utilize domain specific text documents to enrich an existing ontology. Specifically, given a master ontology O_M to be enriched, Text2Onto is first used to build a set of relevant ontologies O_L s from domain specific documents. Then, ontology alignment and merging algorithms are used to integrate O_M and O_L s into a new ontology O_I . For example, if two concepts in O_M and O_L have the same literal name, they may be treated as a unique concept in the new ontology O_I [Euz04]. However, as the content of the new ontology O_I is controlled by the O_L s learned, O_I may also contain many wrong concepts and relations, same as those ontologies learned from scratch by Text2Onto.

In summary, although many systems have been developed for ontology learning from domain specific text documents (e.g., [BS99, FN98, BNC00, Eng03, MNV02, MS00, RT03]), they are still far less mature. Most of them only use shallow NLP tools to process documents and focus on extracting concepts and taxonomic (IS-A) relations. As a result, the ontologies learned are typically lacking in semantics (particularly, missing non-taxonomic relations), and need to be enriched substantially by the human experts before applying into real-world applications.

2.2.2 Ontology Learning from Database Schemas

Besides defining data structure of databases, database schemas also provide a conceptual model of a working domain as a map of concepts and their relationships, similar to the

CHAPTER 2. RELATED WORK

Figure 2.1: The approach developed in the InfoSleuth project for ontology learning from database schemas.

ontologies. Therefore, database schemas become a possible resource for ontology learning (e.g., [Kas99, Joh94, RHO⁺02, SSV02]). For example, Figure 2.1 presents the approach developed by the InfoSleuth research project for ontology learning from database schemas [Kas99]. Initially, concepts with their attributes are identified from the database schema, where tables in the database are the concept candidates and the columns of the tables are the attribute candidates. Then, relations between the concepts are established by considering the foreign key dependencies and the column values which are concepts.

Compared with the previous ontology learning approach, ontology learning from database schemas is relatively easy, as database schemas have already described the target domain in certain level of abstraction. In many cases we can simply use table names as class names and column names as property names for building ontologies. For example, the D2R server² adopts it as the default method to build ontologies for publishing relational databases on the Semantic Web. However, because databases are normally designed for particular tasks that do not need to store all the data in the domain, the coverage of the database schemas and consequently those of the ontologies learned are indeed limited. The learned ontologies have to be enriched for real-world applications, where a set of supplementary methods are developed for this purpose. For example,

²<http://www4.wiwiiss.fu-berlin.de/bizer/d2r-server/>

[Kas99] enriches the learning ontology by analyzing the user’s queries when using the database. New concepts and attributes will be added if they appear frequently in the users’ queries. Also, if some attributes and concepts are not referenced in the queries, they may be dropped from the ontology. In addition, the taxonomic relations between concepts could be improved by using user queries. However, these supplementary methods cannot remove the limitation fundamentally.

2.2.3 Ontology Learning from Machine Readable Dictionary

As machine readable dictionaries (MRDs) can provide a wide range of information about words in terms of their senses and usage notes for semantic domains, which forms the basic units of concepts [Rig98], they also become an important resource for ontology learning. In [SAKY98], the ontology developing environment DODDLE is designed to reuse MRDs for building ontologies for specific domains. Initially, the users provide a set of domain specific terms as input. Then, DODDLE selects additional terms from the MRD and builds a taxonomic structure for these terms, which is called the *initial model*. Finally, the initial model is improved by a set of algorithms and the users to produce the ontology.

WordNet [Fel98] is a special MRD that groups English words into sets of synonyms called *synsets*, provides definitions to the synsets, and links the synsets with a set of semantic relations such as hypernym and meronym. It has been commonly used in many ontology learning systems for word sense disambiguation and taxonomic relation extraction (e.g., [MS00, Hea92, MNV02]). Meanwhile, given the fact that WordNet contains a good coverage of terms and their relations (mainly the taxonomic relations), people have also explored the approach of ontology learning from WordNet directly.

Gangemi *et al.* [GNV03] propose an approach of learning a tourism domain ontology by enriching WordNet with more semantic relations. In particular, synsets are selected

from WordNet as the concepts of this tourism domain ontology. Then, for each term in the synset, human annotators are employed to assign an appropriate sense based on WordNet. Finally, a set of heuristics are used to link these synsets, i.e., building relations. For example, as the gloss of the term *driver*'s first sense (*driver#1*) is *the operator of a motor vehicle*, *operator* is a hypernym of *driver*, and *motor vehicle* has only one sense in WordNet (*motor vehicle#1*), a relation is thus built between *driver#1* and *motor vehicle#1*. The results show that an ontology can be built in acceptable time. However, the meanings of these built semantic relations are not readily understandable, which have to be interpreted by the experts.

In [NP03, NT04], Ian Niles *et al.* exploit the approach of learning ontology from WordNet with Suggested Upper Merged Ontology (SUMO). For example, for building a human health domain ontology, the terms *Child* and *Baby* are selected from WordNet and mapped as subclass concepts of the concept *Human* in SUMO. As SUMO has provided a rich formalized axiomatization and taxonomy, the learning process is mainly to select the relevant concepts from WordNet. Same as [GNV03], the selection of the terms is almost done manually, making it a tedious process. More importantly, the coverage of the learned domain ontology is limited to the terms indexed in WordNet. For concepts that are not included in WordNet, they would not be included.

2.2.4 Ontology Learning from Wikipedia

Wikipedia is a free online encyclopedia written by numerous volunteers. Besides being an encyclopedic reference, there are several properties making Wikipedia a good resource for ontology learning. Firstly, each Wikipedia article is a single web page describing a specific concept. Compared with common documents, the redundant information in the Wikipedia articles is indeed limited. Secondly, Wikipedia has a wide coverage of the universe. Right now, there are more than 13 million articles in Wikipedia, including most

CHAPTER 2. RELATED WORK

important concepts of various domains. The selection of concepts would be easy. Thirdly, all articles are manually listed in particular categories, which provides a rudimentary taxonomic hierarchy among the concepts.

Yago [SKW07] is a light-weight and extensible ontology built with Wikipedia. In particular, concepts are selected if they are unique Wikipedia articles. Then, taxonomic relations are extracted from the Wikipedia Categories. For example, as the article about *Einstein* is listed under the category *Physicist* and the article about *Physicist* is listed under the category *Scientist*, the taxonomic relations (*Einstein*, *is-a*, *Physicist*) and (*Physicist*, *is-a*, *Scientist*) are extracted for the three concepts. Finally, a set of rules are used to extract attributes and non-taxonomic relations of the concepts from the Wikipedia articles. For example, as *1878* follows the word *born* in the article about *Einstein*, a non-taxonomic relation (*Einstein*, *bornInYear*, *1878*) is extracted. Due to the special properties of the Wikipedia articles, the learned ontology has shown a high accuracy. However, given the limited volume of each Wikipedia article, only a few kinds of non-taxonomic relations will be extracted. Currently, Yago has only fifteen kinds of non-taxonomic relations, such as *bornInYear* and *locationIn*, which are those well studied relations in the information extraction field. It requires human experts' helps to add more domain specific semantics into Yago.

DBpedia [BLK⁺09] is a project that extracts structured information from Wikipedia for building a large knowledge base³. Same as Yago, DBpedia identifies concepts if they are unique English articles in Wikipedia. Then, a set of tools are used to extract useful information from the Wikipedia articles as the attributions and non-taxonomic relations. For example, the population information in the infobox will be extracted as the attribute for each country. However, different from Yago, each concept (called entry in the DBpedia project) will be further classified using Yago, UMBEL, and DBpedia Ontology⁴ besides

³This knowledge base is also called a large multi-domain ontology by itself.

⁴The DBpedia Ontology used here is a manually built ontology that consists of 170 classes with shallow taxonomy structure. It should be differentiated from the DBpedia knowledge base.

the Wikipedia Categories, which makes it possible to fulfill the different applications' requirements. Currently, the DBpedia knowledge base includes over 2.6 million entries with a rich description of the entities, covering domains such as geographic information, people, companies, films, music, genes, drugs, books, and scientific publications. The major issue of the DBpedia knowledge base is that it also only contains a few kinds of non-taxonomic relations, similar to Yago. This problem is suffered by the KYLIN system [WW08] which mainly utilizes the infoboxes in the Wikipedia articles for ontology learning as well.

2.3 Ontology Based Information Retrieval

In this thesis, we study a major application of ontology based solutions, namely *ontology based information retrieval*. Before reviewing the state of the art, we first need to clarify what ontology based information retrieval is. Specifically, *ontology based information retrieval is designed to search for items such as documents and images that have been annotated with ontological information* [SPL07]. It should be clearly differentiated from the other ontology related information retrieval task which searches for available ontologies, ontology modules, and ontology's elements, e.g., Swoogle [DFJ⁺04], Sindice⁵ and sig.ma⁶.

2.3.1 Searching With Ontological Annotations

Since the ontological annotations have provided a precise description of each item, the ontology based information retrieval process can become quite simple. The systems search for items annotated with the ontological elements specified in the users' queries, where traditional boolean search model is sufficient for this task.

⁵<http://sindice.com/>

⁶<http://sig.ma/>

CHAPTER 2. RELATED WORK

Figure 2.2: An example to illustrate how SearchMonkey works.

OntoSeek [GMV99] is one of the pioneer ontology based systems for retrieving yellow pages and product categories. In OntoSeek, each document has been annotated with a product ontology. To search for interesting documents, a user just needs to select relevant concepts and relations from the ontology for formulating queries. For example, if a user is looking for red cars, he would select the concept *Car* and the relation *color* from the ontology and formulate a query like *?X isa Car AND Car color Red*. In response, the system returns the matched documents to the user. As no incorrect results are returned, the system's performance is indeed good.

SearchMonkey⁷ is a framework provided by Yahoo for enhanced search results with ontological annotations. However, different from the common ontology based information retrieval systems which emphasize the use of ontological annotations for more precise

⁷<http://developer.yahoo.com/searchmonkey>

CHAPTER 2. RELATED WORK

results, SearchMonkey still adopts traditional keyword based approach for document retrieval. The ontological annotations are only used to help provide highlighted, relevant links and structured data in the result pages, enabling end users to complete tasks faster. An example to illustrate how SearchMonkey works is given in Figure 2.2, where the annotations of a web page are shown explicitly and structurally in the Yahoo’s search result page.

Note that an ontology has also encoded the domain knowledge and human assumptions in an explicit way. Application can thus further utilize such knowledge for enhanced information services, i.e., *ontology based reasoning*. In [GMM03], the TAP search engine starts at one or more matched concepts in the ontology and then performs a breath first search on the ontology for collecting other relevant information. In [HMS⁺05], possible interesting artifacts will be recommended to the users if they are linked to the selected artifact with some semantic relations defined in the ontology.

At the same time, in view that many items may be returned, which do not have the same degree of relevance to the users’ queries, algorithms are also developed to rank the search results. For example, if a query is *?X isa Researcher* and two results A and B are returned, where A is Professor and B is Ph.D. student, Stojanovic *et al.* [SSS03] will think A is more relevant according to a predefined heuristic that *Professor is more relevant to the concept researcher than Ph.D. student*. In contrast, Castells *et al.* [CFV07] will utilize the frequencies of the concepts in the corresponding documents for ranking A and B. If the concept *Researcher* appears more frequently in the documents about B, B may be seen as more relevant to this query.

With the use of ontological annotations for retrieving information, the complex information retrieval problem has been changed to a simple semantic data retrieval problem, where the performance is only evaluated by success or failure but not precision and recall. Correspondingly, instead of developing complicated search models, the major problems

CHAPTER 2. RELATED WORK

Figure 2.3: A screen shot of Goat’s interface.

to be addressed are how to obtain the annotated data, i.e., *data annotation*, and how to effectively present the users’ information requirements, i.e., *query formulation*.

2.3.1.1 Data Annotation

To obtain annotated data, the simple approach is human labelling. People select relevant semantics from the ontology for labelling the data, where many annotation tools (e.g., [BTMS04, BPS⁺05, HWGS⁺06]) have been developed to facilitate this process. For example, Figure 2.3 shows the GUI of the Goat tool [BTMS04] developed at the University of Manchester for helping users to annotate gene products with Gene Ontology (GO) terms. Different from common tools which do not constrain the choice of GO terms in the annotation process, Goat will suggest the most likely appropriate field values to the user based on his previously entered GO terms. Therefore, the use of Goat can make a great reduction in biological inconsistency and a less tedious annotation process on the part of the user.

Besides designing effective annotation tools, social tagging is also adopted, which allows web users to collaboratively add tags to online objects such as bookmarks, photos,

CHAPTER 2. RELATED WORK

and videos [DJZ⁺09]. For example, Delicious⁸, a social bookmarking services for storing, sharing, and discovering web bookmarks, enables end users to tag each web page freely. Compared with tags which are added individually, the tags assigned collaboratively can reflect the intelligent content of the web pages more precisely.

The use of human labelling is indeed effective for capturing the semantics in the data and producing correct ontological annotations, which guarantees the quality of the information retrieved. However, human labelling is considered too labor-intensive to support scalable applications, even though social tagging is available. Automatic annotation techniques therefore become essential for producing annotated data, especially in the Web context,

OWLIR [SFJ02] is an ontology based document retrieval system developed at Johns Hopkins University. To add ontological annotations to the free texts, an information extraction system is first used to extract the key phrases and elements of the documents. Then, a set of predefined rules are used to translate these key phrases and elements into corresponding concepts and relations. In addition, a set of rules are used to infer more semantics from these translated concepts and relations for enriching the annotations of each document.

Compared with texts, the annotation of multimedia data such as images and videos is more difficult, as there are no simple ways to translate these low-level features to the corresponding high-level concepts. In [BBT06], the Multimedia Ontology Manager (MOM) annotation tool is developed that initially clusters a set of video clips as concepts with a Fuzzy C-Means algorithm. Then, new video clips are automatically labelled using the most similar video cluster. In [BF06], Berg and Forsyth build a classifier for image annotation. Four types of cues, including nearby words, color, shapes, and texture, are combined to determine the relevant concept of an image. In contrast, Jeon *et al.* [JLM03]

⁸<http://delicious.com/>

```
SELECT ?capital
WHERE {
    ?capital isa City;
    ?country isa Country;
    ?capital isCapitalOf ?country;
    ?country isInContinent Asia;
}
```

Figure 2.4: The SPARQL query for retrieving all country capitals in Asia.

develop a cross-media relevance model which estimates the conditional probability of observing a concept c given the observed visual content of an image for image annotation.

Automatic annotation techniques have shown their effectiveness in many applications, which produce a large amount of annotated data. The major deficiency of these automatic techniques is that they are designed for handling fixed semantics, which cannot be easily updated in response to the evolution of the ontologies.

2.3.1.2 Query Formulation

In view that the information retrieval problem has been cast into a semantic data retrieval problem, a set of SQL like languages such as SPARQL [PS06] and RDQL [Sea04] are developed for effectively retrieving ontological annotations. For example, Figure 2.4 presents the SPARQL query which searches for the capitals of all the countries in Asia.

These query languages are indeed effective in retrieving ontological annotations. However, they all impose strict requirements on the usage. The relevant semantics, such as the *City* and *Country* in the above example, has to be specified in the queries explicitly. The use of wrong semantics or the missing of semantics both lead to failures in retrieval. Also, the queries need to be formulated with a fixed grammar, which people have to follow exactly. As a result, using these query languages becomes a big challenge

CHAPTER 2. RELATED WORK

Figure 2.5: A screen shot of the SHOE search engine.

for ordinary users.

To tackle this problem, sophisticated user interfaces are developed to guide users to formulate queries. For example, Figure 2.5 presents the screen shot of the SHOE search engine designed for ontology based document retrieval [HH00]. To use it, the user firstly has to choose an ontology from the database. Then, he needs to select the relevant concepts (i.e., categories in the GUI) from the list that best describe the subject of his query. Finally, the system formulates queries according to the user's selections and retrieves documents from the database.

The well designed interfaces do help users to correctly formulate queries. However, they lose the flexibility in query formulation compared with traditional keyword based queries or natural language queries. Alternative approach is thus developed that enables the users to formulate traditional keyword based queries or natural language queries and then infers related semantics from the user's queries automatically. In [KMH04], the user's queries are first translated into a set of tokens by stemming and removing

CHAPTER 2. RELATED WORK

stop words. Then, a list of synonyms is used to map the remaining tokens to particular concepts of the ontologies. Similarly, Contreras *et al.* [CBB⁺04] use natural language processing (NLP) tools to parse the users' natural language queries and then identify the relevant semantics with a predefined thesaurus.

Compare with the above approach of designing user interface, the extraction of related semantics from the user's queries enables the users to have enough initiative in formulating queries. However, it needs a robust NLP tool to process the queries and a large predefined list/thesaurus to map the extracted elements to the semantics in the ontology. For example, if the thesaurus or the domain ontology does not specify that *Washington Wizards* is the same as *Washington Bullets*, the corresponding concept *Washington Wizards* could not be extracted from the query *tell me about team Washington Bullets* [KMH04]. As a result, this approach may not capture the users' information requirements correctly.

2.3.2 Searching With Keywords

Although utilizing ontological annotations is the standard way of retrieving information, there are several reasons to continue supporting keywords in ontology based information retrieval. Firstly, compared with ontological annotations that impose a high cognitive demand on the users, keywords are more intuitive for describing an information resource from the human perspective, helping the users to specify their requirements. Secondly, as mentioned above, it is not that easy to formulate queries for retrieving ontological annotations. In contrast, people have been accustomed to expressing their information requirements in terms of keywords. The well selected keywords are also effective for information retrieval, which do not place any heavy burden on the users and the systems. Finally, there are still many resources which are lack of ontological annotations. The use of keywords is the only way of exploiting those resources.

QuizRDF [DW04] is a search engine that employs both ontological annotations and keywords for document retrieval. Just like the other ontology based information retrieval systems, QuizRDF returns documents whose annotations are the same as those specified in the users' queries. In addition, documents which contain the keywords in the queries are also returned. In fact, there is no strict difference between using ontological annotations and keywords for retrieving documents.

Zhang *et al.* [ZYZ⁺05] develop a fuzzy description logic based model that incorporates keyword based search into ontology based search for enhanced document retrieval. In particular, keyword based queries are issued to search for documents that cannot be retrieved with ontological annotations. Then, the documents retrieved are modelled as a set of fuzzy concepts to represent the queries, whose values are determined by their retrieval status values (RSVs). Finally, the new modelled concepts are used with the existing concepts defined in the ontologies for formulating queries. As such, the system will not be constrained by the coverage of the ontology used.

A more interesting search engine is presented in [RSdA04], which combines keywords with ontological annotations for information retrieval. Specifically, the system first retrieves documents which contain at least one of the keywords specified in the users' queries. Then, the annotated concepts of these retrieved documents are collected as the inputs of a spreading activation procedure, which finally infers the most relevant concept to the users' queries. Compared with the above two approaches, it provides a different way that enjoys the benefits of both keyword based queries and semantic information. However, this system currently only supports the retrieval of concepts but not documents.

2.4 Ontology Based User Modelling

The ultimate objective of an information retrieval system is to understand the needs of each individual and to effectively and knowledgeably address each individual's need in

CHAPTER 2. RELATED WORK

Comic (0.3)
Animated (0.2)
...
Computers (0.5)
Hardware (0.0)
Software (0.5)
...
Shopping (0.6)
Auctions (0.1)
...
Sports (0.8)
Baseball (0.3)
Football (0.1)
...

Figure 2.6: An example to illustrate the user model built in [PG99], where the values assigned to each concept represent the user's degree of interest to this concept.

a given context [Rie00], i.e., *personalized information services*. To achieve this objective, the system requires the understanding of individual's requirements and behaviors, i.e., **user modelling**. Traditional systems, e.g., [TT98], only adopt keywords for user modelling. However, due to the deficiencies of keywords in representing and organizing information, the resultant user profiles cannot represent a user's interests accurately. This results in the poor performance of these systems when providing services. In view of the benefits brought by using ontologies for information representation and organization, people have also begun to exploit ontology based user models for this purpose.

Pretschner and Gauch [PG99] are among the pioneers to make use of ontology for modeling users and for providing personalized document access. Initially, a domain ontology⁹ is employed to organize documents such that each document is classified as a particular concept in the domain ontology. Then, by analyzing a user's surfing history, they obtain the user's interest in each concept and record them as the user model. Finally, personalized document accesses are performed by referring to the stored user's interest

⁹The domain ontology used is the Lycos category (<http://www.lycos.com>) which contains 5,863 concepts and has a tree depth of four for the concept hierarchy.

CHAPTER 2. RELATED WORK

Figure 2.7: An illustration of the distance based inference method.

factor for each concept in the user model. As the user model is structured hierarchically according to the domain ontology (an example is given in Figure 2.6), it is deemed as an ontological user profile. Unfortunately, it is not known whether the other components of the ontology, such as the semantics of the relations and the structure of the ontology, could be used for user modeling. Also, this user modeling method does not consider the change in the user's interests [LMMP96]. The user profile would thus be inaccurate and result in significant degradation in the system's performance over time. A similar ontological model is proposed by Vallet *et al.* [VCF⁺07] for personalized multimedia content access.

Middleton *et al.* [MSR04] present another ontology based user model for information recommendation. The user profile is also built based on the user's browsing history and a pre-defined domain ontology, similar to [PG99]. Furthermore, a distance based inference method is performed when capturing the user's interests in each concept (see Figure 2.7). When a user selects a low level concept in the concept hierarchy, say *Quantum Theory*, it hypothesizes that the user may also be interested in the high level concepts, e.g., *Physics* and *Natural Sciences*. Values are subsequently assigned to the high level concepts to indicate the user's degree of interest, where the values to be assigned are determined by their distances from the low level concept. Finally, academic papers are recommended based on the user's interest factor for each concept recorded in the user model.

Compared with the user model developed in [PG99], Middleton's user model considers the internal structure among the concepts for better capturing a user's interests. However,

a major deficiency of this user model is that the inference method used only considers the distance between two concepts for assigning interest values. Such an approach may be suitable for processing ontologies with taxonomic relations only [MSR04]. However, there are normally other kinds of semantic relations in the ontologies besides the taxonomic relations. For example, the concept *Quantum cryptography* is linked to the concept *Quantum Theory* through the *relatedTo* relation. Simply treating these different relations equally during inference has shown to be inefficient in our prior experiments. More importantly, the semantics of the relations should be used to capture a user’s interests. For example, a user would prefer concepts linked to the concept *Quantum Theory* through the *relatedTo* relation to those through the taxonomic relation, since these concepts are more specific. In this case, the *relatedTo* relation is more important to this user. But the distance based inference method cannot use such information for user modeling.

A variant of Middleton’s user model is adopted by Chirita *et al.* [CNP05] that utilizes a more complex function to determine the interest factors assigned to each concept. However, it still merely uses relations to indicate that certain concepts are connected, and the semantics of the relations is not considered for user modeling. To build more precise user profiles, it is critical to explore effective ways of combining semantic relations with concepts for representing a user’s interests.

2.5 Ontology Analysis

Research on ontology analysis exploits the possible ways of automatically or semiautomatically discovering necessary changes in the ontologies to facilitate a continual evolution of ontologies. Based on the different resources used for this task, we can classify the existing ontology analysis approaches into three categories, namely *ontology driven change discovery*, *usage driven change discovery*, and *data driven change discovery*.

2.5.1 Ontology Driven Change Discovery

Ontology driven change discovery is about finding changes within an ontology itself. Different from the usage driven and data driven change discoveries, the changes discovered from an ontology do not reflect the changes in conceptualization or users' requirements of the ontology. They appear mainly because there are some design flaws in the ontology that need to be fixed.

In [Sto04], a heuristics based approach is adopted to analyze the structure of an existing ontology for finding necessary changes. For example, a heuristics defined as *A concept with a single subconcept should be merged with its subconcept* would remove certain concepts from the ontology. The main idea of this approach comes from that of applying refactoring for object-oriented software architecture [TB99]. In such a way, it can reduce the unnecessary content for maintaining coherence of the ontology.

As a kind of important ontology management tool, the ontology versioning systems are able to discover necessary changes in the ontologies by managing different variants of an ontology and the derivation relations between the variants [KF01]. Its basic idea is similar to that of the concurrent versions system (CVS) for software source code maintenance. However, ontology versioning is more complex. For example, Figure 2.8 presents the two different ways for defining a class in RDF schema. Both are valid and have exactly the same meaning. In this case, simply detecting difference in representation is not sufficient. It further requires the detection of difference in the semantic level.

In [KFKO02], the OntoView tool is developed which firstly materializes all the statements in the ontology and then checks for the modifications of ontologies according to a set of rules. In contrast, the PromptDiff tool [NKKM04] first locates the unchanged parts in the ontology. Then, a set of heuristics are applied to the unchanged parts for finding modifications in the ontologies. In [ZTC07], four different ways are proposed to compare different versions of RDF ontologies by considering RDF Deltas. The results

```
<rdfs:Class rdfs:ID="Car"/>
```

```
<rdfs:Description rdfs:ID="Car"/>
  <rdfs: type rdfs:resource="http://www.w3.org/2000/01/rdfschema#Class"/>
</rdfs:Description>
```

Figure 2.8: The two different ways of defining a class in RDF schema.

have shown the effectiveness of these systems for ontology versioning, which keeps the consistency of ontologies and enables the reuse of ontologies in new situations without invalidating the current usage.

2.5.2 Usage Driven Change Discovery

One of the difficult problems when adopting the ontology based approach is that we do not have any measure that can directly evaluate the quality of a built ontology. Whether an ontology is well designed is only known after employing it for providing services, where the best test is the end users' responses when interacting with the system. The *usage driven change discovery* is thus proposed to analyze the usage information for finding the anomalies in the ontologies and giving clues on how the ontology should be improved [SSGS03].

Oberle *et al.* [OBHG03] present a framework for analyzing the users' navigation patterns when browsing an ontology based semantic portal. Specifically, each URL visited will be mapped to a set of corresponding entities in the ontology, which consequently forms a semantic log file for each user. Then, data mining methods are applied on the semantic log files for finding useful patterns. For example, a close association between the concept *Person* and the concept *Publication* found from the log file could suggest adding a relation between the two concepts.

In [Sto04], both the users' queries and browsing history are utilized for change discovery. Specifically, the users' queries are used to measure the users' degree of interest

CHAPTER 2. RELATED WORK

to each concept. For the most frequent concepts in the queries, they would be considered for decomposing into several subconcepts. For concepts that no body is interested in the queries, they would be suggested for deletion from the ontology. Different from [OBHG03], the user browsing history is used for refining the concept hierarchy. For example, the three concepts *BSc Student*, *MSc Student*, and *Phd Student* are all recorded as subconcepts of *Student* in the original ontology. However, the frequency of visiting the concept *BSc Student* after visiting the concept *Student* is found to be much higher than that of *MSc Student* and *Phd Student* in the browsing history. A new concept *Graduate Student* would be added into the ontology as the parent of *MSc Student* and *Phd Student*.

There are other applications such as the OntoShare system [DDS03] that alters the semantics of concepts based on the users' shared documents and the Bibster system [HHSTS05] which relies on the users' rating scores to change the ontology's elements. The major deficiency of those applications is that they have to wait for sufficient usage data for finding meaningful patterns. As a result, the systems cannot respond to the changes in real time.

2.5.3 Data Driven Change Discovery

Instead of waiting for sufficient usage information for change discovery, the *data driven change discovery* is proposed to discover necessary changes from possible data resources directly. In particular, two types of data resources are used in this approach. The first group is the domain relevant resources such as domain specific text documents and database schemas, which are also used for ontology learning. By monitoring these domain relevant resources, the systems can find necessary changes in the ontologies. For example, Cimiano and Völker [CV05] employ a probabilistic ontology model (POM) that records the distributions of the concepts and relations in the corpus when building ontologies. If new corpus comes with different concepts and relations distributions, it implies that

CHAPTER 2. RELATED WORK

Figure 2.9: An example of the ontology instance data.

certain changes have happened in the ontologies and consequently a set of algorithms are used to find those changes.

The other data resource, which will be exploited in this thesis, is the data that instantiates an ontology, i.e., ontology instance data. For example, Figure 2.9 presents an example of the ontology instance data wherein web pages (i.e., instances) are annotated with relevant concepts in an ontology. Simply, we could count the frequency of the concepts in the annotated web pages for change discovery. If a concept is not used to annotate any web page, it would be suggested for deletion from the ontology [MMS⁺03].

As the population of ontology instance data has involved the end users' perceptions about the ontology, the changes discovered from the ontology instance data can produce the same effect as those found from the user behaviors for reflecting the end users' requirements about the ontology. At the same time, the collection of ontology instance data is much easier. Consequently, ontology instance data driven change discovery becomes a promising approach to supporting ontology evolution.

In [MZ02], Maedche and Zacharias utilize ontology instance data for refining the existing concept hierarchy in the ontology. In particular, a set of similarity measures are defined which consider the taxonomic similarity, relation similarity, and attribute similarity to cluster similar ontology instance data, and further produce hierarchies among the

clusters. Then, the resultant hierarchy is compared with the existing concept hierarchy, which results in possible recommendations for refining the concept hierarchy.

In [Nak02], the inductive logic programming (ILP) method [LD94] is used for extracting meaningful rules from the positive and negative ontology instance examples, which can introduce new concepts into the ontology and put them into appropriate position in the concept hierarchy. It is a quite straightforward but effective way of using ILP method, since ontology instance data have provided the basic statements, i.e., ontological annotations, for inductive learning.

A more recent survey is in [TBRH08], which exploits possible machine learning techniques for change discovery from ontology instance data. As shown in the paper, for performing ontology instance data driven change discovery, the major issue is how to correctly interpret the ontology instance data as the inputs of the existing learning algorithms. In this thesis, we will treat the ontology instance data as single labeled graphs and apply frequent subgraph mining algorithm for discovering necessary changes in the ontologies.

2.6 Summary

In this chapter, we have reviewed the existing research related to ontology learning, ontology based information retrieval, ontology based user modelling, and ontology analysis. In summary, the existing techniques for ontology learning and analysis still require much humans' involvement. At the same time, the developed ontology based information retrieval systems and user models are not sufficiently effective for providing information services. Starting from the next chapter, we shall present a collection of methods and systems developed for addressing these raised issues for ontology learning from domain specific text documents, ontology based information retrieval and user modelling, and ontology instance data driven change discovery.

Chapter 3

Mining Ontological Knowledge from Domain Specific Text Documents

3.1 Introduction

To adopt the ontology based solution for providing information services, we firstly need to create ontologies for representing and organizing information. However, ontology building is known to be a tedious process. Manually acquiring knowledge for building ontologies requires much time and resources, which cannot fulfill our requirement. Ontology learning systems [GPMM] are thus developed to extract concepts, relations between concepts, and axioms on relations from domain specific text documents. However, in the current state of the art, the technologies for tackling this problem are far less mature (see Chapter 2). Most of the existing systems only use shallow (or light) Natural Language Processing (NLP) tools to process documents and focus on extracting concepts and taxonomic (IS-A) relations. As a result, the learned ontologies are typically lacking in semantics (particularly, missing non-taxonomic relations), and need to be enriched substantially by the human experts before applying into real-world applications.

In view of the problems suffered by the existing ontology learning systems, we present a system called Concept-Relation-Concept Tuple based Ontology Learning (CRCTOL) for automatically mining ontologies from domain specific text documents in this chapter.

Specifically, CRCTOL adopts a full text parsing technique and employs a combination of statistical and lexico-syntactic methods, including a statistical algorithm that extracts key concepts from a document collection, a word sense disambiguation algorithm that disambiguates words in the key concepts, a rule based algorithm that extracts relations between the key concepts, and a modified generalized association rule mining algorithm that prunes unimportant relations, for ontology learning. As a result, the ontologies learned by our system are more concise and contain a richer semantics in terms of the range and number of semantic relations compared with alternative systems.

We conduct two case studies where CRCTOL extracts ontological knowledge, specifically key concepts and semantic relations, from a terrorism domain text collection and a sport domain text collection. At the component level, quantitative evaluation by comparing with Text-To-Onto [MS00] and Text2Onto [CV05] has shown that CRCTOL produces much better accuracy for concept and relation extraction. At the ontology level, we employ a wide range of quantitative and qualitative methods, including a structural property based method to verify the quality of the learned ontological network, comparisons to WordNet based on the taxonomic relations extracted, scoring the learned ontology's quality by the experts, and directly comparing with a human-edited benchmark ontology, which all demonstrate that ontologies of high quality are built.

3.2 System Architecture

The CRCTOL system (Figure 3.1) consists of six components, namely *Data Importer*, *Natural Language Processing*, *Algorithm Library*, *Domain Lexicon*, *User Interface*, and *Data Exporter*.

Data Importer: As our system only supports plain text documents, the Data Importer converts documents of other formats, such as PDF, XML or HTML, into plain texts, wherein the structural information of the documents, such as DTD, is discarded.

CHAPTER 3. MINING ONTOLOGICAL KNOWLEDGE FROM DOMAIN SPECIFIC TEXT DOCUMENTS

Figure 3.1: The architecture of the CRCTOL system.

Natural Language Processing: The NLP component incorporates a set of NLP tools, such as the Stanford’s Log-linear Part-Of-Speech Tagger [TKMS03], for tagging words with POS tags and the Berkeley Parser [PBTK06] for identifying the constituents in the sentences and their relationships¹. With the NLP component, we can utilize a full text parsing technique for text analysis. This function distinguishes our system from many alternative systems which only use POS tagging and shallow parsing techniques. An example of the full text parsing technique is given in Figure 3.2, wherein the POS and syntactic tags have been assigned to the sentence.

Algorithm Library: The algorithm library consists of a statistical algorithm that extracts key concepts from a document collection, a word sense disambiguation (WSD) algorithm that disambiguates the key concepts, a rule based algorithm that extracts relations between the key concepts, and a modified generalized association rule mining algorithm that prunes unimportant semantic relations.

Domain Lexicon: The domain lexicon contains terms specific to the domain of interest and their attributes, which are used in the NLP component for analyzing documents.

¹In general, the Berkeley parser can be used directly to process documents. But for special cases where the words have unusual attributes, we first need to train the Log-linear tagger to assign correct POS tags and then parse documents with the Berkeley parser.

CHAPTER 3. MINING ONTOLOGICAL KNOWLEDGE FROM DOMAIN SPECIFIC TEXT DOCUMENTS

<p>Input: The number of persons killed or wounded in international terrorist attacks during 1999 fell sharply because of the absence of any attack causing mass casualties.</p> <p>Output: ((S (NP (NP (DT The) (NN number)) (PP (IN of) (NP (NP (NNS persons)) (VP (VP (VBN killed)) (CC or) (VP (VBN wounded) (PP (IN in) (NP (NP (JJ international) (JJ terrorist) (NNS attacks)) (PP (IN during) (NP (CD 1999)))))))))) (VP (VBD fell) (ADVP (RB sharply)) (PP (RB because) (IN of) (NP (NP (DT the) (NN absence)) (PP (IN of) (NP (NP (DT any) (NN attack)) (VP (VBG causing) (NP (NN mass) (NNS casualties)))))) (. .))</p>
--

Figure 3.2: A sample input sentence and the corresponding output of the NLP component.

For instance, the word *bin* is usually recognized as “NN”², indicating *bin* as a common noun. However, *bin* can also be treated as a proper noun (“NNP”) as it is the name of “Osama Bin Ladin”. The domain lexicon records such information in order to improve the accuracy of the NLP component. It is manually built and can be updated by the user during the learning process.

User Interface: After a domain ontology is built, it will be shown in the user interface (see Figure 3.3). The left panel of this interface lists the concepts of the domain ontology. When a concept is selected, its surrounding information will be shown as a graph in the right panel of the interface. Therefore, the users can easily explore the internal structure of the learned ontology. Furthermore, this interface enables the users to edit the learned ontology by adding or removing concepts and relations between concepts. The concepts and relations extracted incorrectly can thus be removed or corrected by the user.

Data Exporter: In CRCTOL, the learned domain ontology is modeled as a labeled graph, which is a compact and abstract representation of the ontology, and can be easily translated into other knowledge representation languages. The Data Exporter is used to translate the learned ontology into a target representation language. At present, CRCTOL supports two ontology languages, namely RDFS [RDF] and OWL [OWL].

²The instructions of the POS tags can be found in <ftp://ftp.cis.upenn.edu/pub/treebank/doc/tagguide.ps.gz>.

CHAPTER 3. MINING ONTOLOGICAL KNOWLEDGE FROM DOMAIN SPECIFIC TEXT DOCUMENTS

Figure 3.3: The user interface.

The overall procedure for ontology learning is summarized as follows.

- *Data Preprocessing*: Documents of other formats are converted to plain texts before learning ontologies.
- *NLP Analyzing*: Input files are processed using the NLP component. POS and Syntactic tags are assigned to individual words and sentences in the documents.
- *Concept Extraction*: Concepts are extracted and identified by a statistical algorithm from texts. These concepts are called the key concepts in the target domain.
- *Word Sense Disambiguation*: The senses of the key concepts are identified using a variant of the Lesk algorithm [Les86].
- *Semantic Relation Extraction*: The semantic relations of the key concepts are extracted from the text, which include taxonomic and non-taxonomic relations.

- *Ontology Assembling*: An ontology is built in this step by correctly assembling the concepts and relations extracted. The redundant relations are removed. The final ontology is represented in the form of a graph.
- *Ontology Exportation*: The users explore the built ontology with the user interface, modify the ontology if necessary and export the learned ontology.

3.3 Concept Extraction

To build an ontology, the initial step is to find the important concepts of the target domain. As terms correspond to linguistic representation of concepts in the texts [SDM80], concept extraction is thus used to those domain specific terms from texts. In our system, concept extraction consists of two steps. Firstly, possible candidate terms (*a set of lexical units*) are extracted from texts with certain linguistic filters³, i.e., term extraction. Then, domain specific terms are identified from those candidate terms with a particular statistical measure, i.e., term selection. This module plays a key role in the ontology learning process, whose performance greatly affects the system's overall performance for building ontologies.

3.3.1 Concept Extraction Procedure

Ontology learning systems typically adopt one of the following two approaches to extract concepts. The first one, as used by Xu et al. [XKPS02], initially identifies a set of single-word terms, particularly nouns, from the texts as the seed concepts. Then, multi-word terms are formed by combining these single-word terms using certain statistical measures such as Mutual Information measure [Fan61]. As a result, the multi-word terms may not be natural in the texts and are coined merely from the statistical aspect.

³In particular, the linguistic filters are a set of POS tags and Syntactic tags based rules.

CHAPTER 3. MINING ONTOLOGICAL KNOWLEDGE FROM DOMAIN SPECIFIC TEXT DOCUMENTS

The second approach, adopted by Text-To-Onto [MS00] and OntoLearn [MNV02], employs a set of pre-defined linguistic filters (particularly the POS tag based rules) to extract candidate terms, including single-word terms and multi-word terms, from texts. Then, some statistical measures, such as the *tf/idf* [MS00] and the *DR&DC* measure [MNV02], are used to rank the extracted terms. Only terms whose values or ranks are greater than a threshold are selected as the concepts.

In CRCTOL, we follow the second approach for concept extraction. However, from prior experiments, we find that most domain specific concepts are multi-word terms. The small number of relevant single-word terms can either be found appearing frequently in the multi-word terms or easily inferred based on the multi-word terms. But the existing relevance measures, such as the *tf/idf* measure, all prefer single-word terms. In this case, although the extracted concepts are correct, it is hard to enrich the learned ontology. For example, in the terrorism domain, if we have the concept *international terrorist group*, the concept *group* can be included automatically in the ontology if it is missing. On the other hand, it is inconceivable to add the concept *international terrorist group* into the ontology if we only have the concept *group*. Therefore, we consider a different strategy for term extraction, focusing on multi-word term extraction. Single-word terms are added if they appear frequently in the multi-word terms or they are found related to the multi-word terms through certain semantic relations in the texts. Compared with the ontologies learned with the existing approach, our ontologies can be easily enriched.

An example to illustrate the concept extraction procedure in CRCTOL is given in Figure 3.4. The detailed steps are described below.

- (i) Extract all multi-word terms from texts. As concepts are nouns or noun phrases in texts, only lexical units with NP⁴ tag are collected.

⁴“NP” is the tag used in our NLP software to annotate nouns and noun phrases.

CHAPTER 3. MINING ONTOLOGICAL KNOWLEDGE FROM DOMAIN SPECIFIC TEXT DOCUMENTS

Figure 3.4: An example to illustrate the concept extraction procedure, where the multi-word concept *terrorist group* and single-word concept *group* are extracted from texts.

- (ii) Remove articles and descriptive adjectives such as “a”, “many” and “several” from the terms extracted.
- (iii) Group all possible sets of two or more words in each extracted term to form candidate terms. For instance, *terrorist attack* is generated from *international terrorist attack*.
- (iv) For each generated multi-word term t , compute its domain relevance value $DRM(t)$. The $DRM(t)$ score, described below, is a statistical measure for evaluating a term’s relevance to the target domain. Terms with high DRM values are selected to form an initial concept list of the domain ontology.
- (v) Let V be the set of single-word terms appearing in the initial concept list as the syntactic head of a term t . For instance, *attack* is the syntactic head of the term $(NP (JJ terrorist) (NN attack))$. We compute for each single-word term in V its frequency in the initial concept list. Those with frequency above a threshold δ are added to the concept list.

3.3.2 Concept Extraction Measure

3.3.2.1 Review of Existing Relevance Measures

In Text-To-Onto and its successor Text2onto, the tf/idf measure is used to determine the domain relevance of the extracted terms. In particular, given an extracted term t in a document set d , the term frequency (tf) and inverse document frequency (idf) are computed as follows:

$$tf = \frac{\text{count of term } t \text{ in } d}{\text{total number of terms in } d} \quad (\text{Eq. 3.1})$$

$$idf = \log_2 \frac{\text{the size of } d}{\text{count of documents where term } t \text{ appears}} \quad (\text{Eq. 3.2})$$

$$tf/idf = tf \times idf \quad (\text{Eq. 3.3})$$

The original tf/idf is designed for identifying important keywords in individual documents for the purpose of information retrieval [Rij79]. It is, however, not suitable for identifying significant concepts of a text collection. For example, given a domain specific concept t , it may appear in many documents in d as it is popularly used in the domain of interest, i.e., $df(t) \approx |d|$. However, t may not be selected as $idf(t) \approx 0$. In other words, tf/idf 's performance is sensitive to the size of d . It cannot work effectively on data sets with limited number of documents, even if these documents may be very long.

To overcome the deficiency of tf/idf , the KFIDF measure [XKPS02] is proposed that utilizes multiple document collections of different domains for concept extraction. The measure is computed by

$$\text{KFIDF}(w, D_i) = \text{docs}(w, D_i) \times \log\left(\frac{n \times |D|}{|D(w)|} + 1\right) \quad (\text{Eq. 3.4})$$

where $\text{docs}(w, D_i)$ is the number of documents of the particular domain D_i in which a term w occurs, n is a smoothing factor, $|D(w)|$ is the number of different domains in

CHAPTER 3. MINING ONTOLOGICAL KNOWLEDGE FROM DOMAIN SPECIFIC TEXT DOCUMENTS

which w occurs, and $|D|$ is the total number of different domains. Words that have high KFIDF values in D_i will be selected as the concepts of D_i . However, KFIDF only considers the importance of the document frequency for concept extraction. To effectively identify and separate domain specific terms, it requires that the total number of different domains $|D|$ should be large enough, as many terms would have the same KFIDF values with a small $|D|$.

In the OntoLearn system, two statistical measures (DR & DC) [MVF03] are used together to identify domain specific concepts.

Domain Relevance (DR): The domain relevance of a term t in domain D_i is given by

$$DR(t, D_i) = \frac{p(t|D_i)}{\sum_{i=1}^n p(t|D_i)} \quad (\text{Eq. 3.5})$$

where $DR \in [0, 1]$, n is the number of document collections, and the conditional probability $p(t|D_i)$ is estimated as

$$E(p(t|D_i)) = \frac{\text{freq}(t \in D_i)}{\sum_{i=1}^n \text{freq}(t \in D_i)}$$

Domain Consensus (DC): The domain consensus of a term t in domain D_i is given by

$$DC(t, D_i) = H(P(t, d_j)) = \sum_{d_j \in D_i \wedge t \in d_j} p(t, d_j) \times \log_2\left(\frac{1}{p(t, d_j)}\right) \quad (\text{Eq. 3.6})$$

where d_j represents documents in D_i which contain t , and the probability $p(t, d_j)$ is estimated as

$$E(p(t, d_j)) = \frac{\text{freq}(t \in d_j)}{\sum_{d_j \in D_i \wedge t \in d_j} \text{freq}(t \in d_j)}$$

Terms with high DR values and DC values, ranked by a linear combination of DR and DC (i.e., $\alpha \times DR + (1 - \alpha) \times DC$, $\alpha \in [0, 1]$), are selected as domain specific terms.

The above two statistical measures however suffer from the following problems. First, the DR measure does not consider the rare event property of concepts [Dun93]. If we substitute the estimation $E(P(t|D_i))$ back, the DR measure can be written as

$$DR(t, D_i) = \frac{\frac{\text{freq}(t \in D_i)}{\sum_{i=1}^n \text{freq}(t \in D_i)}}{\sum_{j=1}^n \frac{\text{freq}(t \in D_j)}{\sum_{i=1}^n \text{freq}(t \in D_i)}} \quad (\text{Eq. 3.7})$$

After simplifying this formula, we see DR measure is actually computed by

$$DR(t, D_i) = \frac{freq(t \in D_i)}{\sum_{i=1}^n freq(t \in D_i)} \quad (\text{Eq. 3.8})$$

So, in OntoLearn, the *Domain Relevance* value merely depends on the term's frequency in the target domain corpus and the contrasting corpora. If we adjust the size of the target domain corpus or the size of the contrasting corpus, the result will be greatly different.

Also, the DC measure is not suitable for concept extraction. Suppose a term t appears in two documents with a frequency of one in each document and a term s appears in the same two documents with a frequency of two in each document. Using the DC measure, $DC(t) = DC(s)$. This conclusion is not appropriate as terms with a higher occurrence frequency should naturally be more important.

3.3.2.2 Domain Relevance Measure (DRM)

In CRCTOL, we develop a new relevance measure called *Domain Relevance Measure (DRM)* for concept extraction. The ideas behind this measure are presented as follows.

Firstly, we incorporate syntactic information into multi-word term extraction. Although this approach is more effective than that of using POS tag based linguistic filters only, it still suffers from the same problem that these extracted lexical units may not be cohesive enough to be treated as a term. In other words, they may be coined together by chance. Traditional approaches [Dai96, BdRP97, MS00] use statistical measures such as Mutual Information measure and the likelihood ratio test to score the extracted terms for tackling this problem. In our system, we consider using the term frequency tf for this purpose, since the tf measure is simple but has shown to produce better performance than other measures for multi-word term extraction [Dai96].

Secondly, we aim to find domain specific terms. The simple approaches, such as the DR measure used in OntoLearn, only consider the frequency of the terms in different

Table 3.1: Contingency table.

	$\mathcal{A}$	$\overline{\mathcal{A}}$	Total
Frequency of Term t	a	b	$a+b$
Frequency of other Terms	c	d	$c+d$
Total	$a+c$	$b+d$	$a+b+c+d$

document sets for tackling this problem. As a result, their performances are greatly affected by the data sets used and many irrelevant concepts may be selected. To achieve our goal, we consider using the likelihood ratio test [Cas90] which has shown to be statistically reliable for this task.

Here, we consider only a two-class problem, selecting terms from a target domain $\mathcal{A}$ with a contrasting domain $\overline{\mathcal{A}}$. The contingency table of a term t 's frequency in $\mathcal{A}$ and $\overline{\mathcal{A}}$ is given in Table 3.1. Suppose the probabilities of t 's occurrence in $\mathcal{A}$ and $\overline{\mathcal{A}}$ are p_1 and p_2 respectively. The likelihood ratio test verifies the hypothesis that the probabilities of the term t 's occurrence in $\mathcal{A}$ and $\overline{\mathcal{A}}$ have the same value p and is thus written as

$$\lambda(t) = \frac{\max_p p^{k_1}(1-p)^{n_1-k_1} p^{k_2}(1-p)^{n_2-k_2}}{\max_{p_1, p_2} p_1^{k_1}(1-p_1)^{n_1-k_1} p_2^{k_2}(1-p_2)^{n_2-k_2}}, \quad \lambda \in [0, 1] \quad (\text{Eq. 3.9})$$

where k_1 and k_2 are the frequencies of t in $\mathcal{A}$ and $\overline{\mathcal{A}}$, and n_1 and n_2 are the total number of terms in $\mathcal{A}$ and $\overline{\mathcal{A}}$ respectively. Referring to the contingency table, the variables in Eq. 3.9 are computed by $k_1 = a$, $k_2 = b$, $n_1 = a + c$, $n_2 = b + d$, $p = \frac{a+b}{a+b+c+d}$, $p_1 = \frac{a}{a+c}$, and $p_2 = \frac{b}{b+d}$. Terms with low λ values will tend to have distinct occurrence probabilities in the target domain $\mathcal{A}$ and the contrasting domain $\overline{\mathcal{A}}$. They can be used to separate $\mathcal{A}$ and $\overline{\mathcal{A}}$.

Finally, besides the likelihood ratio test, we also think the document frequency of a term in the document set, df , is a good measure for judging a term's relevance to the target domain. If a term a appears in multiple documents, it would be more relevant compared with those appearing a single document, even though a has the same term frequency as those terms.

Figure 3.5: A sample input and the corresponding output of the word sense disambiguation module.

With the above considerations, we develop the *Domain Relevance Measure* for identifying domain relevant concepts. Specifically, given a multi-word term t extracted in the collection $\mathcal{D}$ with the linguistic filters, its Domain Relevance value $DRM(t)$ is computed by

$$DRM(t) = \frac{tf(t)}{\max(tf)} \times \frac{|\log \lambda(t)| - \min|\log \lambda|}{\max|\log \lambda| - \min|\log \lambda|} \times \frac{df(t)}{\max(df)}, \quad (\text{Eq. 3.10})$$

where $\min|\log \lambda|$ is the minimum $|\log \lambda(t)|$ value found, $\max|\log \lambda|$ is the maximum $|\log \lambda(t)|$ value found, and $DRM(t) \in [0, 1]$. Terms with high DRM values are selected as the concepts of the domain ontology.

3.4 Word Sense Disambiguation

After terms are extracted, a WSD algorithm is used to identify the intended meaning of each term in the target domain. A sample input and the corresponding output of this WSD algorithm is given in Figure 3.5, where the senses of the two words *international* and *terrorist* are identified. The results will mainly be used for taxonomic relation extraction.

In CRCTOL, we develop a variant of the LESK algorithm [Les86], called VLESK, for word sense disambiguation. The VLESK algorithm is an automatic and unsupervised method, which can be used in different domains without retraining.

Table 3.2: The senses of cone and pine in the dictionary.

Cone:

1. Solid body which narrows to a point
2. Something of this shape whether solid or hollow
3. Fruit of certain **evergreen trees**

Pine:

1. Kinds of **evergreen tree** with needle-shaped evergreen tree
 2. Waste away through sorrow or illness.
-

3.4.1 LESK Algorithm

As a well-known unsupervised WSD algorithm, the LESK algorithm disambiguates word senses using a context window based on two assumptions. First, if words are close to each other in the sentence, they would be related to the same topic. Second, if they talk about the same topic, their glosses in the dictionary should contain the same words. Lesk [Les86] demonstrates the LESK algorithm for separating the sense of Cone in the term Pine Cone from that of cone in the term Ice Cream Cone.

In the Oxford Advanced Learner’s Dictionary of Current English, there are three senses of Cone and two senses of Pine, as shown in Table 3.2. As the first sense of Pine and the third sense of Cone both contain the same words: evergreen tree, the sense of Cone in Pine Cone is thus the third sense in the dictionary, which is different from its sense in Ice Cream Cone.

3.4.2 WordNet

WordNet [Fel98], like traditional dictionaries, contains terms and glosses. But it differs from traditional dictionaries in many aspects. For instance, terms in WordNet are organized semantically instead of alphabetically. In addition, synonym terms are grouped together in synonym sets (called synset). Each synset represents a particular sense of the term, which may be linked to other synsets by certain semantic relations in WordNet.

WordNet stores terms according to four POS tag categories: Noun, Verb, Adjective and Adverb. In WordNet 2.0, there are 114,648 nouns stored in 79,689 synsets, 11,306 verbs stored in 13,508 synsets, 21,346 adjectives stored in 18,563 synsets, and 4,669 adverbs stored in 3,664 synsets.

The major semantic relations between nouns in WordNet are the *Hypernym* and *Hyponym* relations. If the synset A is linked to synset B using the Hypernym relation, A is a kind of B. For instance, the synset {attack, onslaught, onset, onrush} represents the sense: *(military) an offensive against an enemy (using weapons)*. Since it is linked to the synset {operation, military operation} through the Hypernym relation, we infer that {attack, onslaught, onset, onrush} is *a kind of* {operation, military operation}. The *Hypernym* and *Hyponym* relations are reversible and transitive. In addition, the *Meronym* (has-a part of) and *Homonyms* (is-a part of) relations are also used to connect nouns.

For adjectives and adverbs, the key semantic relations in WordNet are the *Similar* and *Also-see* relations, which link adjective or adverb synsets if the two synsets are semantically similar. For example, Synset {international} is linked to {global, planetary, world, worldwide, world-wide} by the Similar relation in WordNet.

There exists one other kind of semantic relations, called *Attribution* relation, linking adjective synsets to noun synsets in WordNet. If the adjective synset A is a value of the noun synset B, A is linked to B by the Attribution relation. For instance, the synset {domestic} is linked to the synset {domesticity} by the Attribution relation.

3.4.3 VLESK Algorithm

The VLESK algorithm implements the original LESK algorithm with the use of WordNet, similar to the previous works [Voo93, BP02]. Particularly, for a target word, the glosses of its related words in WordNet are concatenated with its own gloss as the input (for noun

CHAPTER 3. MINING ONTOLOGICAL KNOWLEDGE FROM DOMAIN SPECIFIC TEXT DOCUMENTS

Table 3.3: The 18 sense combinations of “*international (ADJ) terrorist(ADJ) attack(NOUN)*”.

1: international(adj)_1 terrorist(adj)_1 attack(noun)_1
2: international(adj)_2 terrorist(adj)_1 attack(noun)_1
⋮
17: international(adj)_1 terrorist(adj)_1 attack(noun)_9
18: international(adj)_2 terrorist(adj)_1 attack(noun)_9

word we utilize the Hypernym, Hyponym, Meronym, and Homonyms relations and we use the Similar, Also-see and Attribute relations for adjectives and adverbs). The sense whose gloss shares most common words with those of the neighbor words is selected. If no sense wins, the target word will be assigned its first sense stored in WordNet, as the first sense is the most frequent one in normal use.

Note that the original LESK algorithm disambiguates each word individually. In VLESK, we adopt a parallel disambiguation approach, which is based on the assumption that the chosen sense for a word depends on the senses of its surrounding words. Particularly, given an extracted multi-word term, all possible combinations of the senses of the words in the term are considered simultaneously. A score is computed for each sense combination based on the number of the same words in the expanded glosses of the words. The highest scoring combination is picked as the most appropriate one and each word in the term is assigned its corresponding sense in the winning combination. For instance, given a term “*international (ADJ), terrorist (ADJ), attack (NOUN)*”, there are two senses of *international (ADJ)*, one sense of *terrorist (ADJ)*, and nine senses of *attack (NOUN)* in WordNet. The 18 sense combinations of the three words are shown in Table 3.3, among which the highest scoring combination is the first combination.

The disadvantage of the parallel disambiguation approach is that the algorithm is computationally intensive. Assuming that there are N words on average in a term and S senses on average per word, there are S^N combinations to be compared. Similar

Figure 3.6: A sample input and the corresponding output of the semantic relation extraction module.

approaches that simultaneously disambiguated all words in a context window are adopted in [AR96, CGG92].

3.5 Semantic Relation Extraction

We extract semantic relations between multi-word terms as well as relations between multi-word terms and single-word terms from a text collection. A sample input and the corresponding output of the semantic relation extraction module is given in Figure 3.6. This module is also critical for the system’s overall performance.

Recall that we assume single-word terms are important concepts if they appear in the extracted multi-word terms or they are linked to the multi-word terms with semantic relations. During semantic relation extraction, we may add certain single-word terms into the ontology if they are linked to the multi-word terms detected through non-taxonomic relations.

3.5.1 Taxonomic Relation Extraction

Taxonomic relations are the most important semantic relations in a domain ontology, the exaction of which has been well studied in the field of lexicon building. A simple method for taxonomic relation extraction is string matching. For instance, *international terrorist organization* is recognized as a Hyponym of *terrorist organization* as they contain the

Table 3.4: The lexico-syntactic patterns used for taxonomic relation extraction.

1. NP_0 such as $NP_1\{, NP_2, \dots, (and or) NP_n\}$	$\text{Hyponym}(NP_i, NP_0);$
2. NP_1 is a kind of NP_0	$\text{Hyponym}(NP_1, NP_0);$
3. $NP\{, NP\}^*\{, \}$ or other NP_0	$\text{Hyponym}(NP, NP_0);$
4. $NP\{, NP\}^*\{, \}$ and other NP_0	$\text{Hyponym}(NP, NP_0);$
5. $NP_0\{, \}$ including $\{NP\}^*\{or/and\} NP$	$\text{Hyponym}(NP, NP_0);$

same syntactic head *terrorist organization*. Another method is using lexico-syntactic patterns to extract taxonomic relations. The CRCTOL system uses a combination of both methods.

3.5.1.1 Extracting through Lexico-syntactic Patterns

The first method utilizes the well known lexico-syntactic patterns [ECD⁺04, Hea92], for taxonomic relation extraction. For instance, the lexico-syntactic pattern “such as” [Hea98] is a popularly used pattern for taxonomic relation extraction. Given a sentence containing the “such as” pattern

$$NP_0 \text{ such as } NP_1\{, NP_2, \dots, (and|or)NP_n\},$$

Hyponym relations (NP_i, NP_0) (for $i = 1$ to n) are extracted from the sentence, where the term NP_i is seen as a kind of term NP_0 .

A total of five lexico-syntactic patterns are used in our system. They are listed in Table 3.4.

3.5.1.2 Extracting through Term Structure

Taxonomic relations can also be extracted based on the term structure through string matching. Several heuristics are developed and described below.

- (i) For terms of the form $[word, head]$, if there is a term $[head]$ in the ontology, establish a taxonomic relation between $[word, head]$ and $[head]$. This method is similar to

string match. However, the sense of the matched words, which is identified by our VLESK algorithm, must be the same both terms. For instance, the sense of word *attack* will be the same in term *terrorist attack* and term *international terrorist attack* if *international terrorist attack* is identified as a kind of *terrorist attack*.

(ii) The semantic relations in WordNet can also be used for taxonomic relation extraction. In CRCTOL, we only use the taxonomic and synonymic relations. There are several cases for utilizing them.

- (a) If both $term_1$ and $term_2$ are in WordNet, and there exists a Hyponym or Hypernym relation between $term_1$ and $term_2$, a taxonomic relation $(term_1, term_2)$ is extracted.
- (b) For terms of the form: $term_1 (word1_1, \dots, word1_n, head1)$, and $term_2 (word2_1, \dots, word2_m, head2)$. If $term_1$ has been found holding the taxonomic relation with $term_0$ in stage (1), and $head1$ is in the same synset as $head2$ in WordNet, a taxonomic relation is extracted for $term_2$ and $term_0$. For instance, if we have got the taxonomic relation $(terrorist\ group, group)$, since the sense of the word *organization* in term *terrorist organization* is in the same synset as that of *group* in term *terrorist group*, we can extract the taxonomic relation $(terrorist\ organization, group)$.

3.5.2 Non-taxonomic Relation Extraction

Same as the conventional approach of learning non-taxonomic relations [GGA⁺02, BOS04, CGR⁺05], we hypothesize that *verbs* indicate non-taxonomic relations between concepts. A semantic relation of the $(Concept, Relation, Concept)$ is thus extracted if its lexical realization can be found from the texts, which is in the form of $(Noun_1, Verb, Noun_2)$, where $Noun_1$ is the subject of *Verb* and $Noun_2$ is the object of *Verb*.

CHAPTER 3. MINING ONTOLOGICAL KNOWLEDGE FROM DOMAIN SPECIFIC TEXT DOCUMENTS

We adopt a rule based method for extracting (*Noun*, *Verb*, *Noun*) tuples from texts, similar to the conventional approach. These tuples are used to represent the non-taxonomic relations between the concepts extracted. The noun and verb terms are identified by the regular expressions below:

$$Noun : (DT)?(JJ)^*(NN|NNS|NNP|NNPS)^+$$

$$Verb : (VB|VBD|VBN|VBZ)^+$$

where JJ represents an adjective, NN, NNS, NNP, and NNPS represent nouns, DT represents an article, and VB, VBD, VBN and VBZ represent verbs.

However, different from the conventional approaches that only utilize POS tagging or shallow parsing techniques for non-taxonomic relation extraction, our rules are based on the parse trees obtained with the full-text parsing technique. These parse trees provide grammatical relations between phrases or words in the sentences, allowing us to find the non-taxonomic relations effectively.

An example to illustrate the difference between CRCTOL and the conventional approach is given below, wherein we extract relations from the sentence: *Muslim terrorist groups in this country launched bomb attacks.*

If we use the POS tag based rules to extract non-taxonomic relations, for example, use the following rule defined in Text2Onto:

$$(NN|NNS)^+ (VBD) (NN|NNS)^+,$$

the relation (*Country*, *Launch*, *Bomb Attack*) will be extracted from the sentence (see Figure 3.7). Although this extracted tuple satisfies the rule, it is wrong in semantic sense.

In CRCTOL, the parse tree is utilized for non-taxonomic relation extraction. By analyzing the parse tree of this sample sentence (see Figure 3.8), we can find the true subject of the verb *Launch* is *Muslim Terrorist Group*. The correct relation (*Muslim Terrorist Group*, *Launch*, *Bomb Attack*) is thus extracted.

CHAPTER 3. MINING ONTOLOGICAL KNOWLEDGE FROM DOMAIN SPECIFIC TEXT DOCUMENTS

Muslim	terrorist	groups	in	this	country	launched	bomb	attacks	.
JJ	JJ	NNS	IN	DT	NN	VBD	NN	NNS	.

Figure 3.7: The POS tags assigned for the sample sentence.

Figure 3.8: The parse tree of the sample sentence.

In fact, the primary motivation behind CRCTOL for using a full text parsing technique is because of its effectiveness for non-taxonomic relation extraction. By adopting this approach, the built ontology contains much more semantics, supporting more advanced applications.

3.6 Ontology Assembling

When concepts and relations are extracted from texts, they are classified and integrated to form an ontology correctly. First, taxonomic relation assembling is performed, which builds the main structure of the domain ontology. Then, non-taxonomic relation assembling is performed to link concepts with other types of semantic relations.

3.6.1 Taxonomic Relation Assembling

Because the taxonomy is transitive, relations that can be derived are removed. For instance, given three relations (terrorist group, group), (international terrorist group,

```

For each leaf concept  $C_i$  in the concept hierarchy tree
Repeat
    merge tuples  $(C_i, R, C_k)$ , where  $C_i$  is the subject of relation  $R$ ;
    identify tuples of  $C_i$ 's sibling  $C_j$  such that  $(C_i, R, C_k)$ ;
    If  $C_i$  and  $C_j$  have the same relation  $R$  with  $C_k$ 
        If tuple  $(C_f, R, C_k)$  exist, where  $C_f$  is a hypernym concept of  $C_i$ 
            the frequency of  $(C_f, R, C_k)$  is its own frequency plus those of  $C_i$  and  $C_j$ ;
        Else create new tuple  $(C_f, R, C_k)$ ,
            whose frequency is the sum of the frequencies of  $C_i$  and  $C_j$ ;
    Until no tuples can be merged;
    prune tuples whose support and confidence values are below the threshold;

```

Figure 3.9: The Pseudo-Code of the non-taxonomic relation integration algorithm.

terrorist group), and (international terrorist group, group), the last relation will be removed.

3.6.2 Non-taxonomic Relation Assembling

We adopt a variant of the generalized association rule mining algorithm for integrating non-taxonomic relations. First, all non-taxonomic relations of concept C_i in which C_i is the subject of the relation (C_i, R, C_k) are collected. Then tuples, which have similar verbs (i.e., the verbs can be found in the same synset in WordNet) and same object concepts, are merged. Finally, only tuples with certain confidence and support values are kept. Note that the setting of the support and confidence values depends on the size of the target domain corpus. The pseudo-code of the algorithm for integrating non-taxonomic relations is presented in Figure 3.9. The resultant ontology is a semantic network representing the discourse universe of the target domain.

3.7 Experiments

We have conducted two case studies in which the CRCTOL system is used to build a terrorism domain ontology and a sport event ontology. For the first case study, docu-

CHAPTER 3. MINING ONTOLOGICAL KNOWLEDGE FROM DOMAIN SPECIFIC TEXT DOCUMENTS

ments of the US state department report “Patterns of Global Terrorism (1991-2002)” are downloaded from the web site of Federation of American Scientists⁵ as the test corpus. The PGT corpus contains a total of 104 html files, each of which is about 1,500 words. For the second case study, the SmartWeb Football data set⁶ is used, which consists of 3,542 English documents. Both case studies use the same contrasting corpora, which are collected from the TREC collection, covering the commercial, computer, energy, and general domains.

As new methods are proposed for the three major tasks of concept extraction, word sense disambiguation, and non-taxonomic relation extraction, we first conduct experiments to evaluate the three individual components’ performance. A subset of the PGT data set is manually annotated as the benchmark data set. The corresponding components of Text-To-Onto⁷ and its successor Text2Onto⁸ are used as the baselines. In addition, a version of the CRCTOL system implemented with the Stanford Parser [KM02] is compared to evaluate the robustness of the proposed methods in handling different full text parsing tools.

Then, we estimate the system’s overall performance by evaluating the quality of the ontologies built from the two text collections. In particular, as no benchmark ontology is available for the PGT data set, we use a set of quantitative and qualitative methods to evaluate the quality of the learned ontology, which include a structural property based method to verify the quality of the learned ontological network, comparing with WordNet based on the taxonomic relations extracted, and scoring the learned ontology by the experts. As for the SmartWeb Football data set, we directly compare our results with the accompanied human-edited benchmark ontology. The details are described in the following sections.

⁵<http://www.fas.org/irp/threat/terror.htm>

⁶<http://www.dfki.de/sw-lt/olp2.dataset/>

⁷Version 1.0, released 09/11/2004, <http://sourceforge.net/projects/texttoonto>

⁸Version Beta 4, released 07/08/2008, <http://ontoware.org/projects/text2onto/>

3.7.1 Component Level Evaluation

3.7.1.1 Concept Extraction

Multi-word Term Extraction. Ontology learning systems with shallow NLP techniques utilize only lexical information for multi-word term extraction. In CRCTOL, we make use of both syntactic information and lexical information. Experiments are conducted to evaluate CRCTOL’s performance for multi-word term extraction against Text-To-Onto and Text2Onto, which use a POS tag based rule defined by the following regular expression:

$$(DT)?(VBG|JJ|JJR|JJS) * (NN|NNS)+$$

where DT is the POS tag to represent an article, JJ , JJR, and JJS are the POS tags to represent adjectives, VBG is the POS tag to represent gerunds, and NN and NNS are the POS tags for nouns.

Documents of the PGT corpus (1991-1994) are used as the test corpus. Manual annotation of the document set identifies 3,269 multi-word terms used as the target list for evaluation. The four components are then used to extract multi-word terms from the texts separately. Their performance, in terms of precision, recall and F-measure, is summarized in Table 3.5. We can see both versions of CRCTOL outperform Text-To-Onto and Text2Onto in the experiment, showing that our approach is effective in multi-word term extraction.

However, further investigations find the poor performance of Text-To-Onto and Text2Onto may also be caused by other factors. For example, although using the above POS tag based rule to extract multi-word terms, Text2Onto just returns those multi-word terms whose words are all nouns. As a result, given the sample sentence: *Some African countries have been the venue for terrorist activity in the past*, the multi-word term *terrorist activity* is not returned, as terrorist is tagged as “JJ”. In addition, the different NLP tools

CHAPTER 3. MINING ONTOLOGICAL KNOWLEDGE FROM DOMAIN SPECIFIC TEXT DOCUMENTS

Table 3.5: The performance of Text-To-Onto, Text2Onto, and CRCTOL in multi-word term extraction.

System	Multi-word Terms Extracted		Precision	Recall	F-Measure
	Correct	Wrong			
Text-To-Onto	2,707	353	88.5%	82.8%	85.5%
Text2onto	988	32	96.7%	30.2%	46.1%
CRCTOL (+ Berkeley Parser)	3,090	32	99.7%	97.4%	98.6%
CRCTOL (+ Stanford Parser)	3,113	62	93.5%	95.9%	94.7%

Table 3.6: The performance of the re-implemented POS tag based rule for extracting multi-word terms from the texts processed by the Berkeley parser and the Stanford parser.

Parser for Processing Texts	Multi-word Terms Extracted		Precision	Recall	F-Measure
	Correct	Wrong			
Berkeley Parser	3,098	188	95.7%	97.1%	96.4%
Stanford Parser	3,089	219	90.3%	95.8%	93.1%

used for processing texts affect the performance. Therefore, we re-implement this defined rule with our NLP tools to extract multi-word terms from the texts for comparison.

The performance of our re-implemented POS tag based rule for multi-word term extraction on the texts processed by the Berkeley parser and the Stanford parser, in terms of precision, recall and F-measure, is summarized in Table 3.6. We can see the performance of the POS tag based rule for multi-word term extraction is indeed not that poor, but is slightly lower than that of CRCTOL under the same conditions. The lower precision scores of the POS tag based rule can be attributed to its deficiency in separating gerunds from verbs' present participle. For example, following the definition, it identifies *harboring representatives* from the sentence *The Government of Sudan persisted in harboring representatives of Mideast terrorist groups* as a term. In contrast, CRCTOL works more effectively, since this sentence can be parsed as *(VP (VBG harboring) (NP (NP (NNS representatives)) (PP (IN of) (NP (NNP Mideast) (JJ terrorist) (NNS groups))))*, which indicates *harboring* should not be extracted.

Domain Concept Extraction. After evaluating the performance for multi-word

Table 3.7: The performance of Text-To-Onto, Text2Onto and CRCTOL in concept extraction.

System	Precision	Recall	Correct Concepts Extracted	
			single-word	multi-word
Text-To-Onto	47.2%	2.1%	51	8
Text2Onto	74.4%	3.3%	88	5
CRCTOL (+ Berkeley Parser)	92.8%	4.1%	24	92
CRCTOL (+ Stanford Parser)	92.0%	4.1%	23	92

term extraction, we use the same corpus to assess the ability of the proposed procedure with the DRM measure in identifying domain relevant concepts. Manual annotation of the 33 documents identifies 496 single-word terms and 2,311 multi-word terms as domain specific concepts. The performance is evaluated in terms of precision and recall of the top K concepts selected.

As we can only control the number of multi-word terms to be selected by CRCTOL, the K value is thus determined by the results of CRCTOL. In our experiments, we set the number of multi-word terms to be selected as 100 and finally there are 125 terms selected by CRCTOL from the documents⁹. Therefore, the number of concepts to be selected by Text-To-Onto and Text2Onto is also set to 125¹⁰.

As shown in Table 3.7, both versions of CRCTOL produce much better performance in identifying domain specific concepts. However, due to the many missing multi-word terms shown in the previous experiments, such results are not sufficient to demonstrate the effectiveness of CRCTOL¹¹. Therefore, we pair the procedures used in Text-To-Onto and Text2Onto with the Berkeley parser and the Stanford parser to extract domain relevant concepts for comparison.

⁹The δ is 2 and both versions of CRCTOL selected 25 single-word terms from the multi-word terms.

¹⁰All the systems use a same stopwords list for experiments.

¹¹Not that there are also some other measures such as the entropy measure and the C/NC measure implemented in Text-To-Onto and Text2Onto for concept extraction. However, the former suffers from the same problem as the DR&DC measure and the latter is only for multi-word term extraction, which make them not suitable for concept extraction. Therefore, we only use the tf/idf measure with Text-To-Onto and Text2Onto for concept extraction.

CHAPTER 3. MINING ONTOLOGICAL KNOWLEDGE FROM DOMAIN SPECIFIC TEXT DOCUMENTS

Table 3.8: The performance of the re-implemented Text-To-Onto and Text2Onto for extracting concepts from the texts processed by the Berkeley parser and the Stanford Parser.

Parser for Processing Texts	Concept Extraction Procedure	Precision	Recall	Correct Concepts Extracted	
				single-word	multi-word
Berkeley Parser	Text-To-Onto	80.0%	3.5%	86	14
	Text2Onto	84.8%	3.7%	89	17
Stanford Parser	Text-To-Onto	76.0%	3.4%	82	13
	Text2Onto	80.0%	3.5%	87	13

Note that Text-To-Onto and Text2Onto do not use the same procedure to generate candidate terms. In particular, given a multi-word term extracted from the texts, for example *international terrorist group*, Text-To-Onto further generates a possible candidate, say *terrorist group*, whereas Text2Onto does not. In our re-implementations, we follow their procedures strictly.

The performance of the re-implemented Text-To-Onto and Text2Onto for concept extraction on the texts processed by the Berkeley parser and the Stanford parser is shown in Table 3.8. Although both yield good results, their performance is still lower than that of CRCTOL. As we have shown the difference in multi-word term extraction is small, the better performance of CRCTOL in identifying domain relevant concepts should be attributed to the effectiveness of the proposed concept extraction procedure with the DRM measure.

Finally, we evaluate the robustness of these concept extraction components in handling data sets with different term and document distributions. In particular, we change the test corpus used by concatenating documents of the same year into a single file. Therefore, we have four documents as the inputs, each of which is very long. The robustness is then evaluated by comparing the performance of the different components on the two corpora.

As the original Text-To-Onto and Text2Onto systems have shown their deficiency in the previous experiment, we use our enhanced re-implementations with the Berkeley

Table 3.9: The performance of CRCTOL compared with those of the re-implemented Text-To-Onto and Text2Onto for concept extraction on the new test corpus.

Parser for Processing Texts	Concept Extraction Procedure / System	Precision	Recall	Correct Concepts Extracted	
				single-word	multi-word
Berkeley Parser	Text-To-Onto	55.2%	2.4%	33	36
	Text2Onto	60.0%	2.7%	34	41
	CRCTOL	92.8%	4.1%	23	93
Stanford Parser	Text-To-Onto	48.8%	2.2%	30	31
	Text2Onto	56.8%	2.5%	33	38
	CRCTOL	95.2%	4.2%	24	95

and Stanford parsers as the baselines in the last experiment. The performance is also compared based on the top 125 concepts extracted by these components.

The performance of CRCTOL compared with those of the re-implemented Text-To-Onto and Text2Onto on the new corpus is given in Table 3.9. We can see that both versions of CRCTOL produce a similarly high level of performance on the two corpora while the performance of Text-To-Onto and Text2Onto degrades greatly in the new corpus. Such results clearly show that the concept extraction component in CRCTOL is more robust in identifying domain relevant concepts from document sets with different term and document distributions.

Discussion. We have presented the performance of CRCTOL for concept extraction. We can see the two versions of CRCTOL produce roughly equivalent performance in the experiments, showing that the influence of the different full text parsing tools used on concept extraction is small. Meanwhile, they both greatly outperform Text-To-Onto and Text2Onto, showing that this component is effective for extracting domain relevant concepts. It is notable that the better performance of CRCTOL is mainly attributed to the effectiveness of the proposed concept extraction procedure with the DRM measure in identifying domain relevant terms. Incorporating syntactic information for multi-word term extraction does not improve the performance of concept extraction greatly.

3.7.1.2 Word Sense Disambiguation

We evaluate the performance of the VLESK algorithm by assigning senses to each word of the 100 multi-word terms extracted by CRCTOL in the previous concept extraction stage. Note that we use the term itself as the context window for disambiguation in the experiment. These single-word terms are thus not disambiguated and have several senses in the ontology.

As the VLESK algorithm performs better if the gloss contains more words and each synset in WordNet also contains some example sentences, we further add these example sentences for word sense disambiguation. Particularly, only nouns, verbs, adjectives, and adverbs in the example sentences are added. Meanwhile, a stop word list is used to remove common words such as *a*, *which*, *that*, and *me* from the glosses.

Note that Text-To-Onto and Text2Onto do not provide word sense disambiguation function. We thus apply two other baseline algorithms in this experiment. The first one is the WordNet *1st* sense Method [MN04] which assigns a word its first sense in WordNet. The other one is the random sense algorithm that assigns the word sense randomly. Then, given the senses predicted by the three algorithms, the performance is computed as the number of the words correctly disambiguated divided by the total number of words disambiguated.

The performance of the three algorithms is presented in Table 3.10. We can see that our VLESK algorithm is much better than the random sense algorithm, since the random algorithm is an exceedingly cheap solution. However, our algorithm does not outperform the WordNet *1st* sense baseline greatly, especially on terms with three words. It is caused by the reason that each word's first sense in WordNet is obtained from a large amount of human annotated text and is the most frequent one. Nevertheless, our algorithm is still the best one.

Table 3.10: The performance of the VLESK algorithm compared with the two baseline algorithms.

Algorithm	Two-Word Terms	Three-Word Terms	All Terms
VLESK	78.6%	87.5%	79.1%
Random	47.3%	75.0%	50.5%
WordNet 1st	71.4%	87.5%	73.7%

It is noticeable that the performance of all three algorithms in our experiments, even the random sense algorithm, are better than those reported results of the LESK-like algorithms, whose average precisions are about 50% on the benchmark data sets [BP02]. Such improvement would be attributed to the target domain we work on and the dictionary we use. First of all, the terms extracted from the terrorism domain documents typically have specific meanings. In addition, WordNet has collected many terms relevant to the terrorism domain and their associated relations can be used for sense disambiguation. For example, the stored hyponym relation between the second sense of act and terrorist attack in WordNet helps to identify the correct sense of act in the term terrorist act. Therefore, our WSD algorithms obtain generally better performance.

3.7.1.3 Semantic Relation Extraction

As all the systems use the similar approach to taxonomic relation extraction, we only compare the performance of CRCTOL with those of Text-To-Onto and Text2Onto in non-taxonomic relation extraction. In particular, we first conduct experiments on simple structure sentences¹². Then, we evaluate the performance on general sentences that include both simple structure sentences and complex structure sentences. Such a setting can clearly show the advantage of our proposed method for non-taxonomic relation extraction.

¹²A simple structure sentence is in the form Subject + Verb + Object, where the Verb comes after its Subject and is followed by its Object, e.g., five terrorists died in the battle. No complex components such as adverbial clause or auxiliary verbs are used. The defined POS tag based rules can thus easily extract the non-taxonomic relations from the texts.

Table 3.11: The performance of Text-To-Onto, Text2Onto and CRCTOL for non-taxonomic relation extraction on simple structure sentences.

System	Precision	Recall	F-Measure
Text-To-Onto	96.7%	22.5%	59.6%
Text2Onto	81.8%	20.9%	51.4%
CRCTOL (+ Berkeley Parser)	93.4%	88.4%	90.9%
CRCTOL (+ Stanford Parser)	96.7%	90.7%	93.7%

Table 3.12: The performance of the POS tag based rules for non-taxonomic relation extraction on simple structure sentences.

Parser for Processing Texts	Precision	Recall	F-Measure
Berkeley Parser	84.5%	72.1%	78.3%
Stanford Parser	86.2%	72.9%	79.6%

Simple Structure Sentences. Documents of the PGT corpus (1991-1997) are used as the test corpus in this experiment. Manually annotation of the documents identifies 111 qualified sentences, containing 129 semantic relations. The four systems are then used to extract non-taxonomic relations from these qualified sentences.

The performance of the four systems, in terms of precision, recall, and F-measure, is summarized in Table 3.11. We can see that both versions of CRCTOL outperform Text-To-Onto and Text2Onto in the experiment, especially on the recall value. However, similar to the previous sets of experiments on concept extraction, the poor performance of the Text-To-Onto and Text2Onto systems for extracting relations from these simple structure sentences may also be due to other factors such as the NLP tool used. Therefore, we further implement the POS tag based rules defined in Text-To-Onto and Text2Onto to extract non-taxonomic relations for evaluating this approach's performance accurately.

The performance of the POS tag based rules for non-taxonomic relation extraction on the texts processed by the Berkeley parser and the Standard parser, in terms of precision, recall and F-Measure, is summarized in Table 3.12. We can see the performance of the POS tag based rule is in fact not that bad under the same conditions. But it still extracts fewer correct relations from the texts.

Table 3.13: The performance of Text-To-Onto, Text2Onto, and CRCTOL for non-taxonomic relation extraction on general sentences.

System	Precision	Recall	F-Measure
Text-To-Onto	91.3%	5.5%	48.4%
Text2Onto	75.0%	0.8%	37.9%
CRCTOL (+ Berkeley Parser)	81.5%	55.5%	68.5%
CRCTOL (+ Stanford Parser)	82.4%	55.3%	68.8%

Table 3.14: The performance of the POS tag based rules for non-taxonomic relation extraction on general sentences.

Parser for Processing Texts	Precision	Recall	F-Measure
Berkeley Parser	70.1%	21.6%	45.8%
Stanford Parser	69.8%	21.3%	45.6%

General Sentences. Documents of the PGT corpus (1991) are used as the test corpus. There are 289 sentences in the documents, containing 380 semantic relations. The four systems are used to extract non-taxonomic relations from these sentences. Their performance, in term of precision, recall, and F-measure, is given in Table 3.13. We can see both versions of CRCTOL outperform the baselines greatly when extracting non-taxonomic relation from the general sentences.

Same as the previous experiment on simple structure sentences, we also pair the POS tag based rules together with the Berkeley and Stanford parsers for non-taxonomic relation extraction. The performance of the POS tag based rules with the Berkeley parser and the Stanford parser on the 289 sentences, in term of precision, recall, and F-measure, is given in Table 3.14. Although all the systems' performance degrades in this experiment, we can see the degradation of the POS tag based rules is extremely great. It extracts much fewer relations from the texts, many of which are wrong.

Discussion. We have reported the CRCTOL's performance in non-taxonomic relation extraction. We can see the influence of the full text parsing tools used on non-taxonomic relation extraction is small, as both versions of the CRCTOL system extract many more relations from the texts, especially from the general sentences. Such

differences are exactly due to the ineffectiveness of the POS tag based rules in identifying the subjects and objects of the verbs. When processing simple structure sentences, the deficiency is not obvious. However, for dealing with complex structure sentences, its ineffectiveness becomes immediately apparent. For example, for the sentence *Sikh extremists probably also were responsible for a bombing in New Delhi in late April that killed three people*, it requires a deep understanding of the content to extract the correct relation (*bombing, kill, people*), which cannot be handled with the POS tag based rules. As a result, systems with the POS tag based rules can extract only a few non-taxonomic relations from the texts, since ordinary documents mainly consist of complex structure sentences. As for the CRCTOL system, the only drawback of utilizing the parse tree for non-taxonomic relation extraction is that it requires more time to analyze the sentence structure and build the parse tree. But such costs are reasonable considering that many more relations can be extracted.

3.7.2 Ontology Level Evaluation

In the previous sections, we report the performance of the three components of the CRCTOL system separately. The experimental results show that these components outperform the corresponding components of Text-To-Onto and Text2onto. In this section, we estimate the CRCTOL system's overall performance by evaluating the quality of the ontologies learned from the two text collections.

3.7.2.1 The Terrorism Domain Ontology

As no benchmark ontology is available for the PGT data set, we use a set of quantitative and qualitative methods to make a more objective evaluation about the quality of the learnt ontology.

Ontology Building. 200 multi-word terms are selected from the initial 11,745 multi-word terms as the domain ontology concepts. A high filtering rate is used due to

Table 3.15: The top ten multi-word terms extracted.

Terms	DRM
terrorist group	0.6153
terrorist attack	0.5729
international terrorism	0.3772
terrorist act	0.2399
terrorist activity	0.1758
terrorist organization	0.1744
state sponsor	0.1647
security force	0.1278
car bomb	0.1006

Figure 3.10: The concept *militant group* extracted and the relations around it.

the requirement of creating a concise target domain ontology. Table 3.15 lists the ten most highly ranked multi-word terms with the DRM values.

Recall that single-word terms can be added into the ontology during the semantic relation extraction stage. Besides the 47 single-word terms found in the concept extraction stage, 144 single-word terms are also added as the concepts of the final ontology, each of which has at least 12 relations linked to the multi-word concepts. After relation assembling, there are 271 semantic relations kept in the ontology.

An example of the extracted concepts and the associated relations is shown in Figure 3.10. We can see that the concept “militant group” is a subclass of concept “group” in

the ontology. It is linked to concepts “authority”, “weapon”, “facility”, and “Pakistan” through non-taxonomic relations “surrender to”, “acquire”, “threaten”, and “base in” respectively. The direction of the links indicates that “militant group” is the subject in these semantic relations.

To maintain a generic solution, our work so far does not discriminate concepts and instances. In fact, the distinction between concepts and instances depends on the task requirement and human judgment. For example, the term “Pakistan” is a concept in the ontology generated and it can also be considered as an instance of the concept “Country”. Most ontology learning systems also do not make such a distinction. The few exceptions include Text-To-Onto [MS00] which employs a predefined thesaurus and Text2Onto [CV05] which implements specific methods for instance discovery.

Structural Property Evaluation. It is known that for established knowledge networks, such as the WordNet and Hindi WordNet, their graph representations, like the one shown in Figure 3.10, hold the *small world* property [RUSB07]. Since our built domain ontology is similar to these knowledge networks, its graph representation should also hold the same property. We therefore can indirectly gauge the quality of the built ontology by measuring whether its graph representation is consistent with that of a *small world* graph. It is more objective than human experts’ judgement and can be easily implemented.

Degree Distribution: An essential characteristic of the *small world* graph is that its degree distribution $p(k)$ of the nodes in the graph follows a power-law distribution.

We present the degree distribution of the built ontology’s graph representation and its log-log plot in Figure 3.11. We see that its degree distribution follows a power-law distribution that is characterized by an exponent $\gamma = -1.1754$, showing this graph is a *small world* graph.

Figure 3.11: The degree distribution $p(k)$ and its Log-Log plot for the domain ontology built on the PGT data set.

Clustering Coefficient: A graph is considered small-world if its average clustering coefficient $\bar{c}$ is greatly higher than that of a random graph constructed on the same node set, whose value is closer to $\frac{1}{N}$ [Wat03].

For our graph, we compute an average clustering coefficient of

$$\bar{c} = 0.5113,$$

which is much greater than that of a random graph on the same node set ($\bar{c} = 0.0026$). This again supports the claim that the learned ontology has the *small world* property.

Paradigmatic Relation Evaluation. Paradigmatic relations, such as synonym relation and taxonomic relation, are patterns of association between lexical units that have high semantic similarities (Rapp, 2002). In this section, we evaluate the paradigmatic relations learned by the CRCTOL system. Specifically, we refer to WordNet to judge the taxonomic relation extracted, which would require less subjective scores.

There are a total of 176 taxonomic relations stored in the built domain ontology. However, some of these relations are not suitable for evaluation as they contain concepts

Table 3.16: Two dimensions of evaluating the quality of the built domain ontology, rated by five human judges.

Goodness of concept (average score)	Noise of relation (average score)
3.48	1.60

that are not recorded in WordNet. For example, the concept *guerilla group* is not in WordNet so that we cannot check whether the taxonomic relation between *group* and *guerilla group* is correct with WordNet. After removing those relations, 28 taxonomic relations are used for assessment.

Among the 28 taxonomic relations, 19 are found in WordNet. Five relations not found in WordNet are judged by the experts as the correct ones for the terrorism domain. An example of such taxonomic relations is one between *terrorist attack* and *bombing*. Only four relations are found to be wrong. The overall accuracy of 85.7% illustrates the accuracy of the system in taxonomic relation extraction.

Human Judgement. Finally, we use a qualitative method designed in ConceptNet [LS04] to assess the quality of the built ontology. Five students are employed for this evaluation. Each student is asked to rate twenty randomly selected concepts of the ontology, where the assessment is performed along the two following dimensions, on a Likert 1 (strongly disagree) to 5 (strongly agree) scale:

- (i) *Goodness of concept.* The students are asked to score whether the selected concept is good enough to be kept in the ontology
- (ii) *Noise of relation.* The students are asked to rate whether the associated semantic relations of the selected concept contain wrong information or nonsensical data.

The results of this experiment are given in Table 3.16 and interpreted as follows. For the goodness of concept, most of the selected concepts are rated good enough as concepts

of the terrorism domain. As for the noise of relation, only a few selected relations are rated as incorrect, showing that a relatively clean ontology has been built. On the whole, the scores indicate that the learned terrorism domain ontology is of good quality.

3.7.2.2 The Sport Event Domain Ontology

Different from the PGT data set, a human-edited benchmark ontology is accompanied with the SmartWeb Football data set, which has defined a set of key concepts and relations in the football event domain. Therefore, we could directly compare our result with this benchmark ontology for the concepts, taxonomic relations, and non-taxonomic relations extracted.¹³ However, as the benchmark ontology is manually built independent of this text collection, not all the concepts and relations defined in the ontology can be found in the data set. Also, certain important concepts and relations of the sport event domain are missed by the benchmark ontology but can be found in the data set.

Concept Extraction. This benchmark ontology consists of 608 concepts, which are represented by 1,007 terms, including single-word terms and multi-word terms, in the texts. For example, the concept “LeagueFootballMatch” is defined to be represented by four terms in the texts, namely *football league match*, *football league game*, *soccer league match*, and *soccer league game*. After removing concepts whose terms do not appear in the documents, there are 429 concepts, represented by 629 terms. Our experiments are then to evaluate how many out of the 629 terms can be extracted from the texts.¹⁴

The experimental results of Text-To-Onto, Text2Onto, and CRCTOL, by setting the top 100, 200, 300, 400, and 500 multi-word terms to be extracted by CRCTOL from the texts, are given in Figure 3.12. Compared with the results on the PGT data set,

¹³As we have shown that the deficiency of the original Text-To-Onto and Text2Onto systems in the previous experiments, we only compare CRCTOL with the re-implementations here.

¹⁴Here, we do not consider the problem of finding synonymic terms, as it is only used to refine the learned ontology but not necessary for learning an ontology. Also, people have well studied this problem and many effectively methods, e.g., [BB04], can be used to solve this problem.

CHAPTER 3. MINING ONTOLOGICAL KNOWLEDGE FROM DOMAIN SPECIFIC TEXT DOCUMENTS

Figure 3.12: The performance of Text-To-Onto, Text2Onto, and CRCTOL for concept extraction with different K values.

the relatively poorer performance of CRCTOL on this SmartWeb Football data set can be attributed to several factors. Firstly, most of the terms are with low frequency and document frequency in the data set. For example, only about 150 out of the 629 terms have a frequency greater than 50 in the 3,542 documents. It is thus more difficult to separate the domain relevant concepts with those irrelevant concepts. Secondly, some concepts are represented by different terms in the documents, for example, the concept *FIFA/Coca-Cola world ranking* is represented by *Coca-Cola world ranking* instead of *FIFA/Coca-Cola world ranking*, and the concept *LeagueFootballMatch* is represented by *league game* instead of *football league game* or *soccer league game*. In addition, some relevant terms of the sport event domain are not recorded in the benchmark ontology but found in the data set, for example, the concept *team manager*. Nevertheless, we can see CRCTOL still outperforms Text-To-Onto and Text2Onto in the experiments and extract enough number of domain relevant terms from the texts.

Taxonomic Relation Extraction. For simplicity, we only evaluate the taxonomic relation extracted based on the top 400 multi-word terms extracted from the texts, as such a setting produces the highest F-Measure value in the concept extraction stage. Also, we do not compare CRCTOL with Text-To-Onto and Text2Onto in this experiment as the three systems use a similar approach for taxonomic relation extraction.

There are 633 taxonomic relations extracted for the 499 terms (99 single-word terms are found appearing frequently in the 400 multi-word terms). After removing relations whose associated terms are not in included the benchmark ontology, there are 200 taxonomic relations used for evaluation.

Firstly, we compute the number of taxonomic relations that can be directly matched with those taxonomic relations defined in the ontology. For example, as *central midfielder* has been specified as a subclass of *midfielder* in the ontology, the found relation is thus judged as correct. In total, there are 87 out of the 200 relations found in the benchmark ontology and the precision is calculated as 43.5%.

Then, we compute the number of taxonomic relations that can be derived from the benchmark ontology. For example, since the relation "*match day is a subclass of period*" can be inferred by relations "*match day is a subclass of TournamentRoundStage*" and "*TournamentRoundStage is a subclass of period*", it is judged as correct as well. Under this condition, another 61 relations are qualified. The longest inference rule used involves 6 relations defined in the benchmark ontology.

With the above results, we can see a total of 148 out of the 200 extracted taxonomic relations are correct and the precision is 74.0%. Such a result demonstrates the effectiveness of our taxonomic relation extraction approach. But comparing with the human-edited benchmark ontology, we find the structure of the learnt ontology is relatively flat. The maximum depth of the learned ontology is 3 (e.g., football match $\mapsto$ match $\rightarrow$ competition), as the system cannot effectively identify the subtle difference between the concepts

```

<rdf:Property rdf:about="&sportevent;inMatch" rdfs:label="inMatch">
  <rdfs:range rdf:resource="&sportevent;Football"/>
  <rdfs:domain rdf:resource="&sportevent;MatchTeam"/>
</rdf:Property>

```

Figure 3.13: The relation *inMatch* defined in the benchmark ontology.

from the texts. This limitation is also shared by other ontology learning systems. As such, human efforts are still required for refining the learned ontology.

Non-Taxonomic Relation Extraction. Finally, we evaluate the performance of Text-To-Onto, Text2Onto, CRCTOL for non-taxonomic relation extraction when setting top 400 multi-word concepts extracted by CRCTOL.

There are 97 non-taxonomic relations defined in the benchmark ontology. By removing relations whose domain/range is literal and relations whose associated concepts cannot be found in the texts, 28 non-taxonomic relations are left as the benchmark.

Different from the previous experiments on taxonomic relation extraction, we cannot directly compare our results against the 28 non-taxonomic relations as their representations are not comparable. For example, Figure 3.13 presents the relation *inMatch* defined in the benchmark. We do not have simple methods to map the verbs of the relations extracted, say *play in*, to this relation. Therefore, we employ one student to manually compare the extracted relations with the 28 benchmark relations.

A total of 2,316 non-taxonomic relations are extracted by CRCTOL from texts. After ontology assembling, 250 relations are kept in the learned ontology. However, it is inappropriate to simply compare all the 250 extracted relations against the benchmark relations, as some extracted relations' subject or object is not defined in the benchmark ontology but they may be correct. Therefore, we only evaluate 126 out of the 250 extracted relations which share the same subject and object as the 28 benchmark relations.

As Text-To-Onto and Text2Onto do not provide solutions for removing unimportant relations, we simply remove relations whose frequency is 1 from their results (Text-To-Onto extracts 2,025 relations and Text2Onto extracts 2,059 relations from the texts). Finally, 46 relations are left for both systems for comparison.

The performances of the three systems for non-taxonomic relation extraction are as follows. For CRCTOL, 87 out of the 126 extracted relations are judged as correct, which can be mapped to nine relations defined in the benchmark ontology. The precision is 69.4%. For Text-To-Onto and Text2Onto, only 14 out of the 46 relations are judged as correct, which can be mapped to three relations defined in the benchmark ontology. The precision is 30.4%. We can see CRCTOL outperforms Text-To-Onto and Text2Onto in non-taxonomic relation extraction again.

3.8 Summary

The contribution of this chapter lies on that we present the CRCTOL system for automatically mining ontologies from domain specific text documents. By using a full text parsing technique and incorporating both statistical and lexico-syntactic methods, the ontologies learned by our system are more concise and contain a richer semantics in terms of the range and number of semantic relations compared with alternative systems. We present two case studies where CRCTOL is used to build a terrorism domain ontology and a sport event domain ontology. At the component level, quantitative evaluation by comparing with Text-To-Onto and its successor Text2Onto has shown that CRCTOL can extract concepts and semantic relations with a significantly higher level of accuracy. At the ontology level, the quality of the learned ontologies is evaluated by either employing a set of quantitative and qualitative methods, or directly comparing with a human-edited benchmark ontology, which all demonstrate the high quality of the ontologies learned.

CHAPTER 3. MINING ONTOLOGICAL KNOWLEDGE FROM DOMAIN SPECIFIC TEXT DOCUMENTS

With the support of the ontology learning system for automatically building ontologies, we are able to easily adopt the ontology based solutions for providing services. In the next chapter, we will present an ontology based user model developed for providing personalized services. In addition, a semantic search engine is introduced that utilizes ontologies with keywords for enhanced document retrieval.

Chapter 4

Learning and Inferencing in User Ontology for Personalized Information Services

4.1 Introduction

In this chapter, we study a major application of ontology based solutions, i.e., *ontology based information retrieval*. Currently, in view of the numerous possibilities enabled by the use of ontologies for providing information services, there have been many ontology based information retrieval systems developed. For example, OntoSeek [GMV99] makes use of the concepts in the ontologies in formulating queries so as to improve the precision of the documents retrieved. Hyvnen *et al.* [HMS⁺05] build a semantic portal for the Finnish museums, where a user can browse the collections with the help of the relations in the domain ontology. In addition, ontologies have been used in many other applications including video retrieval [XZZ⁺08] and image retrieval [FGL08].

However, despite the many ontology based applications developed for information retrieval, relatively few of them [MSR04, PG99, VCF⁺07, CNPK05] are concerned with providing personalized services. Following the use of ontology, an obvious advantage of using ontology based models for user modeling is the support of a richer structure as well as more precise definitions of semantics. Among the prior work, Vallet *et al.* [VCF⁺07]

CHAPTER 4. LEARNING AND INFERENCING IN USER ONTOLOGY FOR PERSONALIZED INFORMATION SERVICES

utilize an ontology based user model with contextual information to provide personalized multimedia content access. Middleton *et al.* [MSR04] exploit an ontological approach to modeling users for recommending online academic research papers. Using a similar approach, Pretschner and Gauch [PG99] build user profiles for information re-ranking and filtering. It is important to note that all of the prior work adopt a shallow approach to exploiting semantic information in the domain ontology. Specifically, they only consider the importance of concepts but not that of relations in capturing the user's interests.

A semantically rich user model and an efficient way of processing semantics are the keys to providing personalized services. In view of the existing limitations, we develop an ontology based user model, called *user ontology*, which has the same level of semantics as the domain ontology. Specifically, the notion of *User Ontology* signifies that the semantic relations as used in the domain ontology can also be used for user modelling. A user ontology is a specialization of the domain ontology by assigning each concept and relation of the domain ontology with a specific value for indicating a user's interests. It is a personalized view of the domain conceptualization and is more comprehensive than the existing types of user models in representing a user's interests in a particular domain.

We develop a set of statistical methods for learning individual user ontologies from an existing domain ontology and adopt a spreading-activation theory (SAT) [And83b] based procedure for inferencing in the user ontology. The proposed user ontology model and the spreading-activation based inferencing procedure have been incorporated into a semantic search engine called *OntoSearch* which originally utilizes ontology with keywords for document retrieval and later is extended for image retrieval. It is a seamless extension of the OntoSearch system with the added advantages of using user ontology. We evaluate the performance of the user ontology model using two real-world document sets, namely the ACM digital library¹ and the Google Directory². The experimental results support the

¹<http://portal.acm.org/dl.cfm>

²<http://www.google.com/dirhp>

efficacy of the proposed user ontology model and the validity of learning and exploiting user ontology.

4.2 User Ontology Model

A user model represents a user's interests in a particular subject domain, which forms the basis of providing personalized services. Building upon domain ontology used in the ontology based information retrieval, we propose that a user model should also be a type of ontology based model that captures all the semantics of individual users' interests in the domain ontology. It should be part of and can be extracted from the domain ontology. We call this model *user ontology*.

Formally, a user ontology model can be defined as a structure

$$\Theta = (\mathbf{C}, \mathbf{R}, \sigma, \theta_c, \theta_r)$$

consisting of

- two disjoint sets $\mathbf{C}$ and $\mathbf{R}$, whose elements c_x and r_{xy} are the *concepts* and *semantic relations* in the domain ontology, respectively,
- a function $\sigma : \mathbf{C} \times \mathbf{C} \rightarrow \mathbf{R}$, which associates a pair of concepts with a particular semantic relation,
- a function $\theta_c : \mathbf{C} \rightarrow [0, +\infty)$ and a function $\theta_r : \mathbf{R} \rightarrow [0, 1]$ which assign weights to concepts and relations in the domain ontology respectively.

An example to illustrate the relationship between the user ontology and the domain ontology is given as follows. Consider the sample domain ontology given in Figure 4.1 that represents a partial conceptualization of the Italian soccer teams. We see that “AC Milan” and “Inter Milan” are Italian soccer teams in different leagues. But this domain

Figure 4.1: A partial domain ontology for the Italian soccer teams.

ontology may be too general for individual’s interests. For example, I am a big fan of the AC Milan team. Therefore, the concept “AC Milan” is more important to me than the concept “Inter Milan”. Meanwhile, joining Champion League is more significant, in my opinion, than joining the Serie A League. The existing user modeling methods [MSR04, PG99, VCF⁺07, CNPK05], however, only consider the importance of the concepts in capturing a user’s interests. A user ontology, on the other hand, can capture all the necessary semantics in the domain ontology for user modeling. Specifically, each concept and relation in the domain ontology will be given a certain value for indicating user’s interests. It is a personalized view of the conceptualization and is more comprehensive than the existing types of user models in representing a user’s interests. The illustration of the user ontology is given in Figure 4.2.

For the purpose of implementation, we use a vector $\mathbf{v} = [v_1, \dots, v_n]$, in which each element v_x stores a user’s long term interest in the concept c_x , and a matrix $\mathbf{M} = [m_{xy}]$, in which each element m_{xy} records the user’s long term interest in the relation r_{xy} and $\sum_y m_{xy} = 1$, to represent the functions θ_c and θ_r respectively.

Figure 4.2: An illustration of the user ontology.

4.3 Spreading Activation Theory

In the field of cognitive science, a popular representation for storing knowledge in long term memory is semantic networks [And76]. In a semantic network, concepts are represented as nodes, which are linked through relations. Information processing in the semantic network typically follows the spreading activation theory, in which the activation value of each and every node spreads to its neighbouring nodes. Given a set of initial inputs to specific nodes of the network, after the spreading activation process, each and every concept in the network will be activated with certain values depending on its relations to the neighbouring nodes. As spreading activation theory has been proven to be efficient for inferencing in semantic networks [And83a, And93], which are structurally similar to the user ontology, it is adopted as a natural choice of inferencing in the user ontology.

An illustration of the spreading activation procedure in a user ontology is given below. Referring to Figure 4.3, the node “Team” is initially activated with an activation value of 1.0. Its activation then propagates across the entire semantic network following the spreading activation procedure. When the network stabilizes, all the nodes will be activated with certain activation values such as those shown in Figure 4.4. Note that the activation value of each node does not depend solely on its distance from the initial

CHAPTER 4. LEARNING AND INFERENCE IN USER ONTOLOGY FOR PERSONALIZED INFORMATION SERVICES

Figure 4.3: Initial stage of the spreading activation process.

Figure 4.4: Final stage of the spreading activation process.

node. For instance, the concept “Serie A” obtains a higher activation value than that of “Inter Milan” following the network configuration, which means “Serie A” is considered as more related to “Team”.

The mechanism of the spreading activation theory is hereby defined formally below. Given a source node x and a destination node y , the activation propagation process follows the formula:

$$I_y(t_{i+1}) = O_x(t_i) \times m_{xy} \times (1 - \alpha), \quad \alpha \in [0, 1] \quad (\text{Eq. 4.1})$$

where $I_y(t_{i+1})$ is the input of node y at time t_{i+1} , $O_x(t_i)$ is the output of node x at time t_i , m_{xy} is the weight of the link between nodes x and y , and α is a decay factor to

represent the energy loss in the spreading activation process. In a simplified spreading activation theory, the output of the node y at time t_i is the input of the node y at time t_i , $O_y(t_i) = I_y(t_i)$. Thus, the entire spreading activation process on the user ontology can be summarized into the following formula:

$$\mathbf{O} = [\mathcal{E} - (1 - \alpha)\mathbf{M}^T]^{-1}\mathbf{I}, \quad (\text{Eq. 4.2})$$

where $\mathbf{I} = [I_1, \dots, I_n]^T$ is the input to the network, $\mathbf{M}$ is the matrix representation of the user ontology whose element m_{xy} is the weight of the relation between concepts c_x and c_y , α is the decay factor, $\mathcal{E}$ is an $n \times n$ identity matrix of order n , and $\mathbf{O} = [O_1, \dots, O_n]^T$ is the final output vector of the spreading activation process, in which O_x is the activation value of concept c_x obtained from the spreading activation process.

4.4 User Ontology Learning

After introducing the user ontology model and the associated inference algorithm, we now present a set of learning methods for assigning weights to concepts and relations in a user ontology according to a user's interests and requirements.

4.4.1 Learning Concepts of Interest

Estimating the interest factor of a user to a concept c_x is relatively straightforward. For each concept c_x in the user ontology, the degree of interest v_x is computed using an iterative formula as follows:

$$v_x(t_{i+1}) = v_x(t_i) \times \delta^{-b} + O_x, \quad (\text{Eq. 4.3})$$

where $v_x(t_i)$ represents the user's long term interest in concept c_x at time t_i , and O_x represents the user's current interest in concept c_x . The decay function δ^{-b} is used to prevent saturation of the interest factor c_x , where δ represents the time interval between

time t_{i+1} and t_i , and $b \in [0, 1]$ is a real-value constant. This decay function is chosen over those used in the existing user models [MSR04] for its correctness demonstrated in numerous experiments of human memory [And83a, And93], which is structurally similar to our user ontology model.

4.4.2 Learning Relations of Interest

Learning relations of interest to a user is similar to learning concepts of interest by assigning interest factors to the relations in a user ontology. The assignment applies to both taxonomic as well as non-taxonomic relations. Initially, an estimated prior value m_{xy}^* is assigned to each element of the matrix $\mathbf{M}$. Then, a typical Bayesian solution [Ive84] is used to compute a weighted average of the prior value and the empirical value iteratively for m_{xy} by:

$$m_{xy}(t_{i+1}) = \frac{a \times m_{xy}(t_i) + \text{freq}(r_{xy})}{a + \sum_y \text{freq}(r_{xy})}, \quad (\text{Eq. 4.4})$$

where $m_{xy}(t_i)$ is m_{xy} 's value at time t_i , $\text{freq}(r_{xy})$ is the frequency of the relation r_{xy} appearing in the information resources which the user is interested in, and $a \in [0, +\infty)$ is a constant to normalize the empirical value and the prior value. A small a implies that the user's long term interest in the relations is dominated by the recent observations. This solution is also consistent with the *fan effect*, an important property of the spreading activation procedure, that the activation value propagating from node x to node y decreases as the number of relations associated to node x increases [And93].

4.4.3 An Illustration

A sample scenario for user ontology learning is given as follows. Suppose Jason is a fan of AC Milan and he wishes to find some web pages that introduce AC Milan's new season in the European league and the Serie A league. Given two web pages selected by Jason, say

CHAPTER 4. LEARNING AND INFERENCING IN USER ONTOLOGY FOR PERSONALIZED INFORMATION SERVICES

one is annotated with three concepts “AC Milan”, “European League”, and “Serie A” and two relations “AC Milan join European League” and “AC Milan join Serie A”, and the other is annotated with two concepts “AC Milan” and “European League” and one relation “AC Milan join European League”, we may parse the two selected web pages to get the related concept and relation information. Suppose the current interest in concept “AC Milan”, O_1 , is 0.6. The frequency of the relation³ “AC Milan join European League” is 2 and that of the relation “AC Milan join Serie A” is 1. We may use such information to update the user ontology. Assuming that Jason’s degree of interest on the concept “AC Milan”, the relation “AC Milan join European League”, and the relation “AC Milan join Serie A” at time t_i are as follows: $v_1(t_i) = 0.5$, $m_{12}(t_i) = 0.5$, and $m_{13}(t_i) = 0.5$.

According to Eq.4.3, v_1 will be updated by

$$v_1(t_{i+1}) = v_1(t_i) \times \delta^{-b} + O_1 = 0.5 \times 0.5 + 0.6 = 0.85,$$

where $\delta^{-b} = 0.5$ is the decay value to reflect the change of Jason’s long term interests during t_i and t_{i+1} .

According to Eq.4.4, m_{12} and m_{13} are computed by

$$m_{12}(t_{i+1}) = \frac{a \times m_{12}(t_i) + freq(r_{12})}{a + \sum_y freq(r_{1y})} = \frac{0.1 \times 0.5 + 2}{0.1 + 3} = 0.66$$

$$m_{13}(t_{i+1}) = \frac{a \times m_{13}(t_i) + freq(r_{13})}{a + \sum_y freq(r_{1y})} = \frac{0.1 \times 0.5 + 1}{0.1 + 3} = 0.34$$

where $a = 0.1$ is the normalization parameter value.

After the update, we obtain the user’s degree of interest in the concepts and the relations at time t_{i+1} , given by $v_1(t_{i+1}) = 0.85$, $m_{12}(t_{i+1}) = 0.66$ and $m_{13}(t_{i+1}) = 0.34$. Compared with the user profile at time t_i , the new profile at time t_{i+1} indicates that

³In our experiments, the frequency of a relation r_{xy} for relation learning, $freq(r_{xy})$, is computed by $freq(r_{xy}) = \sum_{d_j} freq(r_{xy}, d_j)$, where d_j is a web page selected by the user and $freq(r_{xy}, d_j)$ is the frequency of the relation r_{xy} in d_j .

Jason prefers the relation “AC Milan join European League” to the relation “AC Milan join Serie A”. Documents annotated with the former relation can thus be recommended with a higher probability.

4.5 OntoSearch: A Full-Text Search Engine

User ontology can be used in many ways to support personalized information services, including document re-ranking, information filtering, and query expansion. Document re-ranking involves the re-ordering of items returned by a search engine by moving items deemed as more relevant to a user towards the top of the list. Information filtering removes irrelevant items by referring to the user profile and delivers the relevant ones to the user. Query expansion adds related concepts to expand the user’s original query for the purpose of improving the precision of the information retrieved. Here, we introduce a semantic search engine called *OntoSearch* that exploits user ontology model for document re-ranking.⁴

4.5.1 Motivation

A key problem of ontology based information retrieval systems is how to effectively use these available ontologies to find the information required by individual users. To tackle this problem, many prior systems have adopted a simple solution which requires the users to include some forms of semantics explicitly in their queries [GMV99, SFJ02]. For example, a user of OntoSeek [GMV99] would need to identify the corresponding concepts of his/her query terms from the domain ontology. If he/she wants to search for information related to jaguar cars, he/she has to specify the concept “car” in the query so that the system can refer to this concept for filtering irrelevant documents. Also, the

⁴In this thesis, we focus on the use of OntoSearch for enhanced document retrieval. Please refer to [WJCT08] for the details about how to use OntoSearch for enhanced image retrieval.

negative concepts to the queries submitted are utilized for further filtering out irrelevant documents [LZ06, TLZN07, TLZ09].

Compared with the traditional keyword based methods, the explicit semantics in the query certainly improves the system’s performance. However, this approach is rather unfriendly for typical users, since it is usually not straightforward to identify the matching concepts and relations of a query from the domain ontology. Although sophisticated interfaces, e.g., [KPO⁺03], have been developed to help users select concepts and relations from the domain ontology, they ultimately do not relieve users from the burden.

In view of this problem, solutions [CBB⁺04, KMH04] have been developed to extract related concepts directly from a submitted query. For example, the concepts *Team* and *Washington Wizards* are extracted from the query “tell me about team Wizards” for retrieving audio data about NBA team Washington Wizards [KMH04]. However, this approach requires a robust natural language processing (NLP) tool to process the queries and a large thesaurus to map the extracted terms with the corresponding semantics in the ontology. In the above example, if the thesaurus or the domain ontology does not specify that “Washington Wizards” is the same as “Washington Bullets”, the corresponding concept *Washington Wizards* could not be extracted from the query “tell me about team Bullets”. In other words, the user’s queries still affect the system’s performance greatly. In this chapter, the OntoSearch system is developed for relieving the users from specifying the semantics explicitly in the queries with a different approach.

4.5.2 Approach and Assumption

The general principle of the OntoSearch system is to utilize the ontological annotations of the documents for automatically inferring the semantics related to the queries submitted. Similar to traditional keywords based search engines, OntoSearch first makes use of keyword based queries to retrieve an initial set of documents. As each document retrieved

CHAPTER 4. LEARNING AND INFERENCE IN USER ONTOLOGY FOR PERSONALIZED INFORMATION SERVICES

Figure 4.5: The procedure of the OntoSearch system for enhanced document retrieval.

has been annotated with specific semantics, by processing the associated semantics of the documents, OntoSearch derives the relevant semantics of the query submitted and uses it for document re-ranking. Specifically, the relevances of the concepts are determined implicitly in OntoSearch through a spreading activation inference in the domain/user ontology and further used to yield an enhanced search performance.⁵ Compared with applications that directly extract concepts from the queries, such an approach is more robust as it does not depend on thesauri or NLP tools for concept extraction, enabling users to formulate queries freely. A similar approach has been used to infer the most relevant concept upon a user's query in [RSdA04].

4.5.3 System Flow

The procedure of the OntoSearch system for handling search queries is highlighted in Figure 4.5. Similar to using a traditional search engine, a user submits queries consisting of keywords to the system, wherein the corresponding semantic annotation is not required. OntoSearch then returns an initial list of documents L obtained with a keyword based search method. Note that the choice of the specific keyword based search method

⁵The OntoSearch system still works if only domain ontology is used, see experiments. In that case, it is a common search engine and does not provide any personalized services.

The OntoSearch System's Overall Algorithm**Main()**

1. SearchDocument()
2. UpdateUserOntology()

SearchDocument() //search documents relevant to the user's query

3. retrieve documents containing keywords in the user's query q , getting a document list L
4. extract concepts from L that are used to annotate documents in L
5. use the extracted concepts as the input of a spreading activation process on the user ontology
6. get the values assigned to the extracted concepts after the spreading activation process finishes, which represent the user's short term interests in each concept
7. combine the short term interests with the user's long term interests stored in the user ontology to get the user's total interests in the extracted concepts
8. re-rank the documents in L with the user's total interests in the extracted concepts
9. return the document list L to the user

UpdateUserOntology() //update user ontology

10. request the user to select the interested documents in L
11. update user ontology based on the ontological annotations of the selected documents

Figure 4.6: The Pseudo-Code of the OntoSearch system's overall algorithm.

does not greatly affect the final performance of the OntoSearch system. We have experimented with the vector space model [BYRN99] and the pagerank algorithm [PBMW98] for document retrieval. They both obtained satisfactory performance.

As the documents are pre-annotated with the ontological information, we can obtain a set of the associated concepts based on the documents in L . Using these concepts as the seeds to our user ontology, the spreading activation theory [And83b] process infers the concepts that are semantically related to the initial concept set. Then, the conceptual relevance scores, in terms of the concept activations in the user ontology, are combined with the user's long term interests to re-rank the documents before presentation to the user. The relevant documents are subsequently used to update the user ontology. The pseudo-code of the system's overall algorithm is given in Figure 4.6. The details are

presented in the following sections.

4.5.4 Ontological Indexing

We use the classical vector space model to index documents stored in the system. Specifically, a document j is represented by a vector

$$\vec{d}_j = (w_{1,j}^k, w_{2,j}^k, \dots, w_{m,j}^k, w_{1,j}^c, w_{2,j}^c, \dots, w_{n,j}^c), \quad (\text{Eq. 4.5})$$

where m is the number of index keywords in the system, n is the number of index concepts, $w_{i,j}^k$ represents the weight of the keyword k_i in document j , and $w_{i,j}^c$ represents the concept c_i 's weight in document j .

For each keyword k_i , its weight $w_{i,j}^k$ is calculated using the traditional *tf/idf* measure [Rij79]

$$w_{i,j}^k = \text{freq}(k_{i,j}) \times \log \frac{N}{n_i},$$

where $\text{freq}(k_{i,j})$ represents the frequency of the keyword k_i in document j , N is the number of documents in the system, and n_i is the number of documents in which keyword k_i appears.

As there is no standard method available for computing the degree of relevance of the concepts to the annotated documents, for each concept c_i , we use a simple method to determine its weight $w_{i,j}^c$ to the documents. If the concept c_i is specified in the document j , its weight $w_{i,j}^c$ is 1, else its weight is 0. This approach is different from that of pagerank-like algorithms in handling conceptual information [GSBS03] and has shown its effectiveness in our experiments.

4.5.5 Inferencing in User Ontology

In our system, after a query is submitted, a list of documents is retrieved from the database using the keyword based search method. As documents are pre-annotated with

the ontological information, we can obtain a set of the associated concepts besides the documents retrieved. The spreading activation theory is then used to infer the concepts of relevance to the user's query from the associated concept set.

Given the associated concepts together with their frequencies obtained, we form a vector

$$\mathbf{I}_q = [I_{1,q}, I_{2,q}, \dots, I_{n,q}]^T$$

as the input to the spreading activation process. Specifically, the input to the concept c_i for a query q is calculated by

$$I_{i,q} = \frac{freq(c_i)}{\sum_{c_i} freq(c_i)}, \quad (\text{Eq. 4.6})$$

where $freq(c_i)$ represents the frequency of the concept c_i in the initial list.

Upon receiving the input $\mathbf{I}_q$, the spreading activation procedure is first performed on the matrix $\mathbf{M}$ to infer the concepts' current relevance to the user's query q by

$$\mathbf{O}_q = [\mathcal{E} - (1 - \alpha)\mathbf{M}^T]^{-1}\mathbf{I}_q.$$

Then, for each concept c_i , the current relevance score $O_{i,q}$ is combined with the user's long term interest v_i to derive a final score $S_{i,u}$. This final score, representing a balance between the user's long time interest and the current relevance to the concepts, is used to re-rank the documents. In our application, the final score $S_{i,u}$ for concept c_i is computed by

$$S_{i,u} = O_{i,q} + v_i \times \delta^{-b}, \quad (\text{Eq. 4.7})$$

where δ represents the time interval since the last query, $b \in [0, 1]$ is a real-valued constant, and δ^{-b} simulates the decay function occurred in the long term memory [And93].

4.5.6 Similarity Measures

Two similarity measures, namely the cosine measure and the position based measure, are used to re-rank documents in the initial list. Those documents deemed as more relevant are moved towards the top of the final list for presentation to the user.

4.5.6.1 Cosine Measure

Similar to the way documents are indexed in OntoSearch, we can represent a query q by a vector

$$\vec{q} = (w_{1,q}^k, w_{2,q}^k, \dots, w_{m,q}^k, s_{1,q}, s_{2,q}, \dots, s_{n,q}), \quad (\text{Eq. 4.8})$$

where $s_{i,q}$ (the normalized value of $S_{i,u}$) represents the relevance of the concept c_i , and $w_{i,q}^k$ represents the keyword k_i 's relevance to the query q . In OntoSearch, the value of $w_{i,q}^k$ is calculated by

$$w_{i,q}^k = \frac{\text{freq}(k_{i,q})}{\sum_{k_j \in q} \text{freq}(k_{j,q})}, \quad (\text{Eq. 4.9})$$

where $\text{freq}(k_{i,q})$ represents the frequency of keyword k_i in the query q .

Now, given the document vector $\vec{d}_j$ and the query vector $\vec{q}$, the similarity measure of a document j to the query q is computed as:

$$\text{sim}(j, q) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \times |\vec{q}|}. \quad (\text{Eq. 4.10})$$

This formula is a classical measure used in the vector space model. Although several variants are available, we adopt this version in OntoSearch as it outperforms the others in our prior experiments. Documents holding a higher cosine value are then moved towards the top of the final list and returned to the user.

4.5.6.2 Position Based Measure

For documents annotated with just one concept, we can use a simple but fast measure for re-ranking. Specifically, we compute a balance value B for the document j by

$$B_j = \beta \times F_j + (1 - \beta) \times T_j, \quad (\text{Eq. 4.11})$$

where F_j is the rank of the document j in the initial list L , T_j represents the rank of the document j 's associated concept among the elements of $\vec{S}_u$, and $\beta \in [0, 1]$ is a constant.

This similarity measure is based on a simple hypothesis that documents holding a higher ranking in L and annotated with the more activated concepts would be more relevant to the submitted query q . Documents with smaller B_j values are then moved towards the top of the final list for presentation to the user.

After browsing the final list, the user may specify those relevant documents, which can be used to update the concepts and relations of interest in the user ontology.

4.6 Experiments

Two real-world document collections are used in our experiments, namely the ACM digital library and the Google Directory. The details are given in the following sections.

4.6.1 Searching ACM Digital Library

4.6.1.1 The Data Set and Domain Ontology

The ACM digital library (<http://portal.acm.org/dl.cfm>) is an online database containing more than 54,000 computer science related articles from 30 journals and 900 proceedings of the Association for Computing Machinery. Compared with ordinary document collections, one key advantage of the ACM digital library is that the publications have been annotated with terms according to the ACM Computing Classification System (CCS)⁶. The CCS can be used as a simple domain ontology which provides a hierarchical structure to describe the various research fields in computer science. Documents indexed using the CCS terms are similar to web pages annotated with ontologies. An illustration of a document in the document set is given in Figure 4.7. We see that this document contains the keywords, including “Bayesian network”, “Markov blanket”, and “classifier”. Also, it is annotated with the concept “G.3 Probability and Statistics”.

As the CCS ontology only contains hierarchical relations, we hypothesize that two concepts are related semantically if they are used to index the same paper so as to enrich

⁶<http://www.acm.org/about/class/1998>

Figure 4.7: An illustration of a document and its annotations in the ACM digital library data set.

the CCS ontology with more semantic relations. For example, the concept *F.2.2 Non-numerical Algorithms and Problems* is related to the concept *G.2.2 Graph Theory* as they are both used to index the Brinkman’s paper [BC05] on dimensional reduction. This relation can therefore be added into the CCS ontology and further used to annotate this paper. Our approach of finding semantic relations between concepts is similar to the one for finding co-citation information in an author analysis application [HHF03].

To facilitate experimentation, we build a local database to store the documents of the ACM digital library⁷, wherein each document is indexed following the approach described in Section 4.5.4. Note that we have adopted an explicit, non-embedded annotation scheme to link a given domain ontology to the documents in the system. For each document, we create a separate file to store the concepts and the relations, and bind it with the original file. Although this method may fail to associate the semantic markup with the specific components in the document, it is relatively simple and easy to implement [SFJ02].

4.6.1.2 Results

A group of ten users is involved in evaluating the user ontology’s ability for providing personalized service.

⁷We mainly download documents published after 1998, since most of the older papers are not labeled with the CCS ontology.

The profile of each user is initialized as follows. For each concept c_x of the user ontology, its value v_x equals zero. For each relation r_{xy} , its initial weight m_{xy}^* is given by

$$m_{xy}^* = \frac{freq(r_{xy})}{\sum_y freq(r_{xy})}, \quad (\text{Eq. 4.12})$$

where $freq(r_{xy})$ represents the frequency of the relation r_{xy} in the data set.

Each user submits two sets of queries to the system, one for training the user ontology and the other for testing. When training the model, one has to browse the top 30 documents returned and provide feedback on the documents that are relevant. This approach is simple but effective in capturing a user's interests. In general, we could also use other approaches to obtain the documents of relevance, for example, by using the surfing behaviors [hCPCP01] or referring to the context environment [RM00].

After training, the learnt user ontology is incorporated into OntoSearch, accordingly called OntoSearch_U for distinguishing from the standard OntoSearch system, to provide recommendation for the test queries. For each test query, the corresponding user has to verify the documents returned and identify their relevance. To conduct a formal evaluation, we further compare OntoSearch_U with three other systems, measuring the performance gain one can obtain with the use of domain and user ontological information.

The baseline system is the Lucene search engine⁸, which employs the keyword based vector space model. The second system is the standard OntoSearch system that utilizes the domain ontology and keywords for document retrieval. Both systems do not have any personalization capability.

Note that OntoSearch_U employs both concepts and relations in the user ontology model. To evaluate the benefit of concepts and relations individually, we conduct experiments on a scaled-down version of OntoSearch_U, called OntoSearch_C, that uses only concept learning in updating the user ontology models. Comparing the performance

⁸<http://lucene.apache.org/>

CHAPTER 4. LEARNING AND INFERENCING IN USER ONTOLOGY FOR PERSONALIZED INFORMATION SERVICES

Figure 4.8: The average precision of OntoSearch_U compared with Lucene, OntoSearch, and OntoSearch_C based on the top five documents retrieved.

of OntoSearch_C and OntoSearch evaluates the benefit of user modeling with concepts. Benchmarking OntoSearch_U and OntoSearch_C investigates the advantage of using relations to capture the users' interests.

The performance of the four systems, namely Lucene, OntoSearch, OntoSearch_C and

Figure 4.9: The average precision of OntoSearch_U compared with Lucene, OntoSearch, and OntoSearch_C based on the top ten documents retrieved.

Table 4.1: The average precision and standard deviation of Lucene, OntoSearch, OntoSearch_C and OntoSearch_U on the top five and ten documents retrieved.

Search Engine	Top Five		Top Ten	
	Avg Prec.	Std	Avg Prec.	Std
Lucene	0.48	0.23	0.51	0.14
OntoSearch	0.58	0.19	0.53	0.16
OntoSearch_C	0.62	0.23	0.59	0.16
OntoSearch_U	0.74	0.19	0.65	0.16

OntoSearch_U, in terms of the average precision for the top five ($Precision@5$) and top ten ($Precision@10$) documents retrieved, is depicted in Figure 4.8 and Figure 4.9 respectively. We can see OntoSearch_U consistently outperforms, or at least produces equivalent performance, compared with the other three systems in the experiments. Based on the top five documents retrieved, OntoSearch_U outperforms the other three systems in five out of the ten users. For the case of top ten documents retrieved, it outperforms in six out of the ten users. The average precision scores of the four systems and their standard deviations are summarized in Table 4.1. We can observe that OntoSearch_U produces the best results among the four systems, validating our approach of using user ontology to enhance search performance in the Semantic Web.

Furthermore, to verify whether the performance gain by OntoSearch_U is statistically significant, we perform paired t-tests on the precision scores over the ten users. The *null hypothesis* is that the performance of OntoSearch_U is equal to those of Lucene, OntoSearch, and OntoSearch_C in retrieving documents. The *alternative hypothesis* is that the performance of OntoSearch_U is better than those of the other three systems. As indicated by the p -values for the paired t-tests reported in Table 4.2, we can see the performance of OntoSearch_U is significantly better than those of the other three systems statistically. In particular, given a Type I Error (α) of 0.05, we can reject the null hypothesis and conclude that OntoSearch_U certainly outperforms OntoSearch_C in terms of precision scores. It follows that semantic relations are useful in capturing the

Table 4.2: The p -values for the paired t-tests on the ACM digital library data set.

Top n	OntoSearch_U over Lucene	OntoSearch_U over OntoSearch	OntoSearch_U over OntoSearch_C
Top Five	0.0002	0.0002	0.005
Top Ten	0.0001	0.0009	0.005

Level 1: Computers

Level 2: Computers/Software

Level 3: Computers/Software/Internet

...

Level 7: Computers/Software/Internet/Client/WWW/Browsers/Firefox

Figure 4.10: An illustration of the ODP taxonomy's structure, where each level represents a particular category.

users' interests.

4.6.2 Searching Google Directory

4.6.2.1 The Data Set and Domain Ontology

The Google Directory (<http://www.google.com/dirhp>) is a service provided by Google that integrates Google's search technology with Open Directory pages (<http://www.dmoz.org/>) for finding information of high quality on the Web. Similar to the ACM digital library, web pages indexed in the Google Directory are labelled using the concepts of the Open Directory Project (ODP) taxonomy⁹.

The ODP taxonomy is a human constructed and maintained taxonomy that organizes the Open Directory pages into 16 main categories and numerous sub categories. An illustration of the ODP taxonomy's structure is given in Figure 4.10. This diagram presents a partial structure of the computer category, where the concept *Firefox* is the lowest category in this structure.

Whereas the CCS ontology only contains taxonomic relations, the ODP taxonomy already encodes many non-taxonomic relations, for example, the *related* relation and the *symbolic* relation. We therefore do not need to use additional methods to enrich this

⁹This taxonomy can be downloaded from <http://rdf.dmoz.org/rdf/structure.rdf.u8.gz>.

taxonomy. But a major challenge of using the ODP taxonomy is that it is very large, containing too many concepts and relations. If we select only the top level concepts, they will be too general to capture a user's interests [CNPk05]. On the other hand, if we use the whole taxonomy, the inferencing process will be very time consuming. After initial experimentation, we select concepts from level 1 to level 3 to form the domain ontology. Altogether, there are 5,965 concepts, 24,506 taxonomic relations, and 52,662 non-taxonomic relations used in our experiments¹⁰.

As the Google Directory contains many more documents¹¹, we do not store all the web pages into a local database. Instead, for each query, we extract the relevant documents with their associated concepts from the Google Directory's result page during run time. They are then re-ranked by the various search systems and returned to the users.

4.6.2.2 Results

A group of ten users is involved in evaluating the user ontology. The setting of this experiment is similar to the one used for the ACM digital library data set. Each user prepares two groups of queries, one for training and the other for testing. When training a user ontology, the corresponding user would browse the top 20 documents returned by his queries and provide relevance feedback to the search engine. The selected documents are then used to update the user ontology.

A problem of training user ontology in the Google Directory data set is that each document is only annotated with one concept. As a result, no explicit links between concepts can be used for relation learning. To solve this problem, we hypothesize that, for those selected relevant documents, if they are labelled with different concepts, relations will be built between these concepts. For example, when searching information about

¹⁰For those documents labeled with the low level concepts, we will use the corresponding high level concepts instead.

¹¹Google Directory has collected over 1.5 million URLs on November 30th, 2008. This latest data is available in <http://www.google.com/dirhelp.html>

CHAPTER 4. LEARNING AND INFERENCING IN USER ONTOLOGY FOR PERSONALIZED INFORMATION SERVICES

Figure 4.11: The average precision of OntoSearch_U compared with Google Directory, OntoSearch, and OntoSearch_C based on the top five documents retrieved.

“grape wine”, documents annotated with the concepts “Recreation/Food/Drink” and “Shopping/Food/Beverage” can both be selected. A semantic relation is thus built for the two concepts and used subsequently for relation learning.

Figure 4.12: The average precision of OntoSearch_U compared with Google Directory, OntoSearch, and OntoSearch_C based on the top ten documents retrieved.

After training, the learnt user ontology is used to provide recommendation for the test queries. As in the previous experiments, we use four systems, namely Google Directory, OntoSearch, OntoSearch_C, and OntoSearch_U, to make comparisons. The performance of the four systems for the ten users, in terms of the average precision for the top five ($Precision@5$) and top ten ($Precision@10$) documents retrieved, is depicted in Figure 4.11 and Figure 4.12 respectively. We observe that OntoSearch_U is the best system of all based on both evaluation criteria. Notably, based on the top ten documents retrieved, it outperforms the other three systems in five out of the ten users and produces equivalent performance to OntoSearch_C for the remaining users. The average precision scores and the standard deviations of the four systems are given in Table 4.3. Although the performance difference is not very significant on the Google Directory data set, OntoSearch_U still produces the best performance, demonstrating the user ontology model's efficacy in supporting personalized services.

As in the previous experiments, we conduct paired t-tests on the precision scores over the ten users to verify whether the performance gain by OntoSearch_U is statistically significant. The *null hypothesis* is that the performance of OntoSearch_U is not different from those of the other three systems in document retrieval. The *alternative hypothesis* is that OntoSearch_U outperforms the other three systems. The p -values for the paired t-tests are reported in Table 4.4. Given a Type I Error (α) of 0.05, we can reject the null hypothesis again and conclude that semantic relations are certainly useful for user modeling.

4.6.3 Discussion

4.6.3.1 Influence of the Ontology's Size on the Performance

In the previous sections, we have reported the performance of the OntoSearch system with user ontology on two real-world document sets. Two key observations can be drawn from

CHAPTER 4. LEARNING AND INFERENCING IN USER ONTOLOGY FOR PERSONALIZED INFORMATION SERVICES

Table 4.3: The average precision and standard deviation of Google Directory, OntoSearch, OntoSearch_C and OntoSearch_U on the top five and ten documents retrieved.

Search Engine	Top Five		Top Ten	
	Avg Prec.	Std	Avg Prec.	Std
Google Directory	0.48	0.09	0.36	0.08
OntoSearch	0.52	0.08	0.46	0.09
OntoSearch_C	0.59	0.07	0.49	0.07
OntoSearch_U	0.62	0.08	0.52	0.08

Table 4.4: The p -values for the paired t-tests on the Google Directory data set.

Top n	OntoSearch_U over Google Directory	OntoSearch_U over OntoSearch	OntoSearch_U over OntoSearch_C
Top Five	0.0002	0.0016	0.0478
Top Ten	0.000002	0.0022	0.0172

the experimental results: 1) OntoSearch systems with the personalization capability (i.e., OntoSearch_U and OntoSearch_C) outperform their counterpart (i.e., Lucene, Google Directory, and standard OntoSearch) without such a capability; 2) OntoSearch with the full set of concept learning and relation learning (i.e., OntoSearch_U) outperforms its counterpart with only concept learning (i.e., OntoSearch_C). Such results support our approach of using user ontology to enhance search performance in the Semantic Web. By recording a user's degree of interest in both concepts and relations, a more precise user profile is built, which can be used subsequently to improve the system's performance for document retrieval.

However, we note that the performance gain brought by relation learning differed rather significantly on the two document collections. As reported in Table 4.5, we can see the precision improvements for the ACM digital library are more considerable than those for the Google Directory. This difference would be caused by the limitation of the user ontology model on large ontologies. Note that a user ontology model is supposed to record the user's degree of interest in the relations besides the concepts. For small ontologies, such as the CCS ontology used in the ACM digital library, this operation is

CHAPTER 4. LEARNING AND INFERENCE IN USER ONTOLOGY FOR PERSONALIZED INFORMATION SERVICES

Table 4.5: Analysis of the performance gain brought by relation learning on the ACM digital library data set and the Google Directory data set.

Data Set	<i>Precision@5</i>			<i>Precision@10</i>		
	OntoSearch_C	OntoSearch_U	Ratio	OntoSearch_C	OntoSearch_U	Ratio
ACM Digital Library	0.62	0.74	19%	0.59	0.65	10%
Google Directory	0.59	0.62	5%	0.49	0.52	6%

Figure 4.13: The average precision of OntoSearch_U on the top ten documents retrieved from the ACM digital library data set with different decay values for inferencing in the user ontology.

relatively easy to perform, as there are fewer relations encoded. But large ontologies normally contain many relations. For example, the trimmed ODP taxonomy used for the Google Directory data set has a total of 77,168 relations. Given the small number of training queries, it is difficult to construct a user ontology to record a user's interests fully on all the relations. Therefore, the improvement on the Google Directory data set is not as significant in our experiments. To guarantee the effectiveness of the user ontology model for capturing users' interests, a longer training time is required for working with large ontologies.

Table 4.6: The average precision and standard deviation of Onto_U on the top ten documents retrieved from the ACM digital library data set with different decay values for inferencing in the user ontology.

α	0.3	0.5	0.8
Avg Prec.	0.65	0.64	0.63
Std	0.16	0.17	0.21

4.6.3.2 Using Customized Parameter Values for Individual Users

Additional experiments are conducted to investigate the possibility of improving the user ontology model's performance by using customized values for key parameters, such as the decay rate α during inferencing, for individual users. As shown in Table 4.6, the average precision of OntoSearch_U on the top ten documents retrieved from the ACM digital library data set does not differ significantly with the different decay values. However, for individual users (see Figure 4.13), some (e.g., user 1 & 7) obtain better results with a large decay value while others (e.g., user 9 & 10) prefer a small decay value. Such results show that setting customized parameter values for individual users is a possible approach to improving the user ontology model's performance. However, adopting such an approach would also suffer from other problems. For example, the complete ODP taxonomy contains more than 590,000 concepts and 1,000,000 relations¹². Finding an appropriate decay value for individual users on this ontology will be extremely time consuming. Consequently, the performance gain obtained may not outweigh the computational cost of finding appropriate parameter values for every user. This is an issue worthy of future investigation.

4.7 Summary

The contributions of this chapter are summarized as follows:

¹²The statistics are based on the snapshot of the ODP taxonomy on 2007-01-25.

CHAPTER 4. LEARNING AND INFERENCING IN USER ONTOLOGY FOR PERSONALIZED INFORMATION SERVICES

- An ontology based user model, called *User Ontology*, is presented for supporting personalized services in the ontology based information retrieval applications. Compared with prior works, the user ontology model has a richer structure and more precise definitions of semantics by utilizing both concepts and semantic relations in the domain ontology for representing a user's interests. Also, we develop a set of methods for learning individual user ontologies from an existing domain ontology and a spreading-activation procedure for inferencing in the user ontology.
- A semantic search engine, called *OntoSearch*, is developed that exploits ontological knowledge with traditional keyword based queries for enhanced document and image retrieval. Compared with alternative systems, the user is not required to specify ontological information explicitly in the queries for retrieving documents. He only needs to consider suitable keywords based on the desired content and the system will infer the relevant concepts automatically. As such, the user can have enough flexibility when formulating queries.

We have integrated the user ontology models into the OntoSearch systems for providing personalized document retrieval and obtained encouraging initial results. In the next chapter, we will utilize the frequent subgraph mining techniques to ontology analysis for enabling up-to-date services.

Chapter 5

Mining Ontology Instance Data For Change Discovery

5.1 Introduction

In this chapter, we focus on the problem of ontology instance data driven change discovery. In particular, given a set of ontology instance data, e.g., the one shown in Figure 5.1, we aim to find the changes in the ontologies for supporting ontology evolution. For example, since the concept *AsianCupMatch* is not used to annotate any web page in the example, it may be deleted from the ontology.

Different from the existing approaches (see Chapter 2), we view the ontology instance data as a single labeled graph (see Figure 5.2), and exploit the frequent graph pattern mining technique for change discovery. For example, the two frequent graph patterns found (see Figure 5.3) in the above example not only indicate that the concepts *ItalianClubTeam*, *Player*, *LeagueMatch*, and *InternationalMatch* are closely connected in the data but also show how they are connected. With such information, we can easily discover the changes in the ontologies.

However, simply mining frequent graph patterns in a single labeled graph has shown to be ineffective for change discovery in prior experiments. Taking the two frequent graph patterns shown in Figure 5.3 as an example, although both pattern A and pattern B appear frequently in the input graph, we can see pattern A is held by all instances of the

CHAPTER 5. MINING ONTOLOGY INSTANCE DATA FOR CHANGE DISCOVERY

Figure 5.1: An example of the ontology instance data, where web pages are connected with hyperlinks and annotated with relevant concepts.

Figure 5.2: A labeled graph corresponding to the example ontology instance data, where the labels are the concepts used for annotating the web pages.

CHAPTER 5. MINING ONTOLOGY INSTANCE DATA FOR CHANGE DISCOVERY

Figure 5.3: The frequent graph patterns found in Figure 5.2.

concept *ItalianClubTeam* while pattern B is only by one instance of this concept. Therefore, if we merely consider the frequency of the graph patterns in the data, an incorrect relation may be added between the concepts *ItalianClubTeam* and *InternationalMatch*. For effective change discovery, it further requires that the frequent graph patterns are also generic enough to represent most instances of the concepts.

A different view of the input graph and the graph patterns found gives us insight of tackling this problem. Referring to Figure 5.2, we can see both pattern A and pattern B are frequent in this labeled graph. However, pattern A distributes globally in the entire graph while pattern B appears only in the right corner as highlighted. Consequently, if we further consider the distributions of these graph patterns in the labeled graph as a criteria, i.e., *mining globally distributed frequent subgraphs*, only pattern A will be identified, satisfying our requirement.

In fact, the above change discovery problem is a particular application of the general *Mining Globally Distributed Frequent Subgraphs in a Single Labeled Graph* problem. For simplicity, we name such graph patterns as **G-Patterns** in this chapter. Those discovered G-Patterns can provide global information about the entire graph and are valuable for a variety of applications such as web usage mining [FCJ01], hypertext classification [CDI98, AW06], and biological data analysis [GSY07, KÖ08]. In the rest of this chapter, we will elaborate more on this general mining problem.

CHAPTER 5. MINING ONTOLOGY INSTANCE DATA FOR CHANGE DISCOVERY

Figure 5.4: The illustration of a single labeled graph G , its subgraph SG , and a graph pattern P identified in G .

5.2 Preliminary Concepts

In this section, we introduce the preliminary concepts to the G-Pattern mining problem.

Definition 5.1 *Labeled Graph*

A labeled graph can be represented by a 4-tuple, $G = (V, E, L, l)$, where

- V is a set of vertices,
- $E \subseteq V \times V$ is a set of edges,
- L is a set of labels,
- $l : V \cup E \longrightarrow L$, l is a function assigning labels to the vertices and edges.

A subgraph of a graph G , SG , is a graph such that $V_{SG} \subseteq V_G$ and $E_{SG} \subseteq E_G$, and the assignment of endpoints to edges in SG is the same as in G .

A labeled graph extracted from the Biozon database (<http://www.biozon.org>) and one of its subgraphs are shown in Figure 5.4, where *Protein*, *Uni_encodes*, *Unigene*, *DNA*, *Uni_contains*, *Uni_encodes*, and *Encodes* are the labels of the vertices and edges. With

this graph, researchers can easily explore relationships between these biological resources. Note that ID:xxx in this example is the ID of the vertex or edge in the database. They are used here for illustration purpose and not included in the patterns mined. The frequent graph patterns¹ that we search for are like the one shown in Figure 5.4c.

Note that we focus on undirected labeled simple graph in this chapter. However, it is easy to extend our algorithm for processing other graphs. For example, we can treat the direction of the edges as labels for processing directed graphs.

Definition 5.2 *Graph Instance*

Let G be a single labeled graph and P be a graph pattern we search for in G , a subgraph of G , SG , is an instance of P in G if there exists an isomorphism between P and SG .

For example, when searching for the graph pattern P shown in Figure 5.5a in the labeled graph G presented in Figure 5.4a, we can see there are four subgraphs of G isomorphic to P (see Figure 5.5b). They are called the instances of P in G .

Definition 5.3 *Edge-Disjoint Based Instance Graph*

*Given the instances of a graph pattern P found in a labeled graph G , we can construct a new graph for P where the vertices represent P 's instances in G and edges are added between two vertices if their corresponding instances share an edge in G . This new graph is called **edge-disjoint based instance graph**.*

For the example pattern P shown in Figure 5.5a, as its instance $i1$ shares the edge $DNA(ID:215)-Uni_contains(ID:62)-Protein(ID:103)$ with $i2$, and $i3$ shares the edge $Protein(ID:78)-Uni_encodes(ID:31)-Unigene(ID:150)$ with $i4$, its edge-disjoint based instance graph is constructed as the one shown in Figure 5.6a. Note that the instance graph is an unlabeled graph. The labels $i1$, $i2$, $i3$, and $i4$ shown in the example are only used to help indicate the relations between the vertices with their corresponding instances.

¹To avoid confusion, we will call the output as frequent graph patterns instead of frequent subgraph patterns in rest of this chapter, since the term *subgraph* has been used in the previous definition for different purpose.

CHAPTER 5. MINING ONTOLOGY INSTANCE DATA FOR CHANGE DISCOVERY

Figure 5.5: An illustration of a graph pattern P and its four instances in the labeled graph G shown in Figure 5.4(a).

Figure 5.6: The edge-disjoint and vertex-disjoint based instance graphs of P .

Besides the edge-disjoint based instance graph, we can also form other types of instance graphs, for example, vertex-disjoint based instance graph, i.e., edges are added between two vertices of the instance graph if their corresponding instances share vertices in the labeled graph (see Figure 5.6b), or even a distance l based instance graph, i.e., edges are added between two vertices if their corresponding instances are reachable with paths of length l in the labeled graph. Forming different instance graphs would be seen as a way of defining the *global distributed* property, since how patterns are called globally distributed are changed in different applications. In other words, the G-Pattern mining

CHAPTER 5. MINING ONTOLOGY INSTANCE DATA FOR CHANGE DISCOVERY

Figure 5.7: An example showing that the downward closure property does not hold for mining a single labeled graph.

problem can be expressed as a function $f(l, \theta)$, where l is the preferred distance between the instances and θ is the minimum support threshold.²

In this chapter, we only consider the edge-disjoint based instance graph³. This setting specifies that *graph patterns are called globally distributed when their instances do not share common edges in the labeled graph*. However, it is trivial to fit all our observations, methods, and conclusions to other conditions.

Definition 5.4 *The Downward Closure Property*

Given a measure f , downward closure property requires that for every pair of patterns p and q in the data set such that P is a subset of Q ⁴ ($P \subseteq Q$), the value of P computed by f is not lower than that of Q , $f(P) \geq f(Q)$.

The downward closure property is essential for the computational tractability of most frequent pattern discovery algorithms, since it can be used to quickly narrow the search space [AS94]. However, when mining single labeled graphs, the standard way of counting the frequency of a graph pattern in the input graph as its support value (i.e., counting the number of instances in the input graph) may not have the downward closure property.

²Given the same support threshold, setting a larger l will lead to fewer patterns to be found in an input graph. Therefore, the number of patterns found with the vertex-disjoint based instance graph would be smaller than those found with the edge-disjoint based instance graph.

³For simplicity, we call it **instance graph** in the rest of this thesis.

⁴We also call Q a child of P in this chapter.

For example, given the two graph patterns P and Q shown in Figure 5.7, while P is a subset of Q ($P \subseteq Q$), Q has a higher frequency than that of P in the input graph G ($\text{freq}(Q) = 15 > \text{freq}(P) = 6$). This indicates that the downward closure property no longer holds. As a result, the algorithm has to search the entire candidate space for finding all the frequent patterns, which results in the mining process time-consuming.

5.3 G-Measure

In this section, we give the details how we address the G-Pattern mining problem. In particular, an objective measure called **G-Measure** is firstly developed that can be incorporated into existing algorithms for identifying G-Patterns. Then, in view that the standard G-Measure does not have the downward closure property for mining G-Patterns, we further develop an approximate method of computing G-Measure for keeping the computational efficiency.

5.3.1 Background

We aim to find globally distributed frequent subgraphs in a single labeled graph. To achieve our goal, the straightforward approach is *divide and conquer*. That is, to partition the input graph into smaller manageable segments, and to find frequent patterns within those segments. However, this approach will suffer from the following problems.

- (i) The quality of the obtained segments is not ensured, as we have no knowledge of the input graph in advance. As a result, the wrong patterns may be mined from the input graphs. For example, if we adopt the k -way partition method [ARK06] to split the input graph, too many regionally distributed patterns would be extracted due to a high k value.
- (ii) More importantly, a certain graph patterns would be lost as their instances happen to occur across different segments. To guarantee the completeness of the G-Patterns

to be found, additional methods are required to recover these lost graph patterns, which would make the mining task even harder.

Alternatively, objective measures are supposed to be developed which can be incorporated into existing frequent subgraph mining algorithms for mining G-Patterns.

5.3.2 Basic Idea

The basic idea of the G-Measure comes from the observations of forming the instance graph: when the instances of a graph pattern appear in the same regional area of the input graph, sharing edges, their corresponding vertices in the instance graph would be directly connected with edges. On the other hand, if these instances are located in different areas, their corresponding vertices in the instance graph are less likely to be linked. For the extreme cases, the instance graph could only contain a set of disconnected vertices, since no common edges are shared by these instances.

An illustration of our observations is shown in Figure 5.8, where we will build the instance graph of P with respect to G . We can see that P 's instances are located in two particular areas of G as highlighted. Following the definition, we build P 's instance graph as shown in Figure 5.9. For instances located in the same areas, their corresponding vertices in the instance graph are closely connected, even forming clique, while for instances distributed in different areas, their corresponding vertices are with a few connections.

Our observations are similar to those happening in the social network where humans of the same class closely connect while humans of different classes are with a few interactions. Therefore, instead of partitioning the input graph, we could test whether a pattern is a G-Pattern by partitioning its instance graph, where the *G-Measure* is to calculate the number of partitions obtained. Such an approach can avoid the problems suffered by the *divide and conquer* approach.

Figure 5.8: The graph pattern P and the input graph G .

Figure 5.9: The built P 's instance graph.

5.3.3 Definition

Before giving the formal definition of the G-Measure, we first present the criterion used for partitioning the instance graph.

Basically, to effectively find community structure in a social network, we assume that for the result obtained, there is a higher density of edges within the same partition than between different partitions. To quantify this requirement, people have proposed different quality measures such as the min cut [SW97] and ratio cut [WS03]. Here, we adopt the *modularity* measure [CNM04] which is commonly used in the social network research field to tackle this problem.

The general principle of the *modularity* measure is presented as follows. Let A_{vw} be an element of this network's adjacency matrix where

$$A_{vw} = \begin{cases} 1 & \text{if vertices } v \text{ and } w \text{ are connected,} \\ 0 & \text{otherwise.} \end{cases} \quad (\text{Eq. 5.1})$$

and suppose this network has been divided into several partitions such that vertex v is in partition P_v . Thus, the fraction of edges that connect vertices in the same partition is

$$\frac{\sum_{vw} A_{vw} \delta(P_v, P_w)}{\sum_{vw} A_{vw}} = \frac{1}{2m} \sum_{vw} A_{vw} \delta(P_v, P_w), \quad (\text{Eq. 5.2})$$

where $\delta(P_v, P_w)$ is 1 if $P_v = P_w$ and 0 otherwise, and $m = \frac{1}{2} \sum_{vw} A_{vw}$ is the number of edges in the network.

Note that for a random graph, the probability of an edge existing between vertices v and w is $k_v k_w / 2m$, where k_v and k_w are the degrees of the vertices v and w . If the social network is correctly partitioned, its value computed by Eq. 5.2 should be greater than that of a random graph. Otherwise, the two values will be the same. Therefore, the *modularity* measure, M , is defined by

$$M = \frac{1}{2m} \sum_{vw} [A_{vw} - \frac{k_v k_w}{2m}] \delta(P_v, P_w). \quad (\text{Eq. 5.3})$$

A higher M value indicates that the network is well partitioned.

With the *modularity* measure, we give the formal definition of the G-Measure as follows:

Definition 5.5 *G-Measure*

The G-Measure is a metric that reflects the distribution of a graph pattern's instances in the input graph. Given a graph pattern P , the G-Measure $G(P)$ is computed as the number of partitions found on P 's instance graph with the modularity measure. If the instance graph is composed of disconnected components, the G-Measure is computed as the sum of the different partitions found in each component.

The definition of the G-Pattern is as follows:

Definition 5.6 *G-Pattern*

Given a graph pattern P and a minimum threshold θ , P is a G-Pattern if and only if $G(P) \geq \theta$.

5.3.4 Conditions of Downward Closure Property

In the previous section, we present the details of the G-Measure developed for tackling the problem. Consequently, we shall incorporate the G-Measure into the existing algorithm for mining G-Patterns. However, as shown in Section 5.2, the standard frequent subgraph mining algorithms may not have downward closure property for mining a single labeled graph. If we simply incorporate the G-Measure into the existing algorithms for mining G-Patterns, many uninteresting graph patterns would be identified and the computation cost is prohibitively high. It further requires that the G-Measure should effectively identify G-Patterns and automatically remove spurious patterns during the mining process. That is, the G-Measure should have the downward closure property.

To test the downward closure property of the G-Measure, we introduce three types of operations occurring in the instance graph. We show that the instance graph of a graph pattern Q is constructed and can only be constructed under the three operations on that of its ancestor P . Therefore, an instance graph based support measure, e.g., the G-Measure, is shown to have the downward closure property if its value is non-increasing under the three types of operations.

5.3.4.1 Three Types of Operations Occurring in the Instance Graph

Given a graph pattern P and its child Q that is with a growth of one edge⁵, Q 's instances cannot occur in the input graph suddenly. They all should be generated from

⁵For simplicity, we only give examples that Q is with a growth of one edge from P in this chapter.

Figure 5.10: An example of the forming clique operation, where the vertices are highlighted with black color.

Figure 5.11: The sample patterns and the input graph corresponding to the forming clique operation shown in Figure 5.10.

P 's instances. The corresponding operations shown on P 's instance graph fall into three categories, namely *forming clique*, *adding edge*, and *removing vertex*.

Definition 5.7 Forming Clique

Given a pattern P 's instance graph $G_P = (V_P, E_P)$ and an arbitrary vertex k on G_P , the *forming clique operation* on k adds a clique $K = (V_k, E_k)$ into G_P , which results in the instance graph $G_Q = (V_Q, E_Q)$ where

$$V_Q = V_P \setminus \{k\} \cup V_k$$

and

$$E_Q = (\{u, v\} | u, v \in V_P \setminus \{k\}, \{u, v\} \in E_P) \cup (\{u', v'\} | u' \in V_P \setminus \{k\}, \\ v' \in V_k, \{u', k\} \in E_P) \cup (\{u'', v''\} | u'', v'' \in V_k, \{u'', v''\} \in E_k)$$

Figure 5.10 presents an example of the *forming clique* operation, where a clique K_3 is added into the instance graph. It happens when several instances of Q are generated

Figure 5.12: An example of the removing vertex operation, where the vertex is highlighted with black color.

Figure 5.13: An example of the adding edge operation, where the vertices are highlighted with black color.

from the same instance of P . The sample patterns and the input graph corresponding to this example are given in Figure 5.11.

Definition 5.8 Removing Vertex

Given the instance graph $G_P = (V_P, E_P)$, removing vertex operation removes a vertex v from the instance graph, which results in the instance graph $G_Q = (V_Q, E_Q)$, where

$$V_Q = V_P \setminus \{v\},$$

and

$$E_Q = E_P \setminus \{\{u, v\} | u \in V_P \setminus \{v\}, \{u, v\} \in E_P\}$$

Definition 5.9 Adding Edge

Given the instance graph $G_P = (V_P, E_P)$, adding edge operation adds a new edge $\{u, v\}$ into the instance graph, resulting in the graph $G_Q = (V_P, E_P \cup \{\{u, v\}\})$.

The examples corresponding to the *removing vertex* and *adding edge* operations are given in Figure 5.12 and Figure 5.13 respectively. The former operation occurs if an instance of P cannot further generate Q 's instance with one edge growth and the latter

CHAPTER 5. MINING ONTOLOGY INSTANCE DATA FOR CHANGE DISCOVERY

Figure 5.14: The sample patterns and the input graph corresponding to the adding edge operation shown in Figure 5.13.

Figure 5.15: The sample patterns and the input graph for illustrating how the instance graph of Q is constructed from that of P with the three operations.

happens when the instances of Q formed share edges in the input graph. The sample patterns and the input graph corresponding to the example *adding edge* operation are shown in Figure 5.14.

5.3.4.2 Constructing Instance Graphs with the Three Types of Operations

After introducing the three types of operations occurring in the instance graph, we now present how the instance graph of Q can be and only be constructed from that of P with them.

First, we show that the instance graph of Q can be constructed from that of P with the three types of operations. An example is given in Figure 5.15, where we will build the instance graph of Q based on that of P (see Figure 5.16a). Firstly, for P 's instances that cannot generate instances of Q with one edge growth, we remove their corresponding vertices from the instance graph (see Figure 5.16b). Then, the *forming clique* operation

Figure 5.16: The detailed steps of building Q 's instance graph.

is performed on the instance graph (Figure 5.16c). Finally, we add a new edge to the instance graph, obtaining the instance graph of Q (see Figure 5.16d). Note that the orders of the operations do not affect the final instance graph obtained. They can be arranged freely when building Q 's instance graph.

Second, we prove that the instance graph of Q can only be constructed from that of P with the three types of operations. For this purpose, we first introduce two other types of operations on the instance graph, namely deleting an edge from P 's instance graph (Figure 5.17) and adding completely new vertices (Figure 5.18) into Q 's instance graph which are not generated from P 's instances. The two types of operations may also be supposed to happen in P 's instance graph when forming Q 's instance graph⁶. Therefore, if the two types of operations exist, our statement is incorrect.

Then, we show that the two operations cannot exist. For example, if the deleting edge operation happens (see Figure 5.17), the corresponding instances of $v1$ and $v2$ in Q 's instance graph, Q_{v1} and Q_{v2} , will not share a common edge in the input graph⁷. However, Q 's instances are generated based on P 's instances. If Q_{v1} and Q_{v2} do not

⁶Given an instance graph, the possible operations on it can only be adding edge, adding vertices, deleting vertices, and deleting edges.

⁷This proof given here is based on the edge-disjoint based instance graph only. However, it is trivial to demonstrate its correctness on other types of instance graphs.

CHAPTER 5. MINING ONTOLOGY INSTANCE DATA FOR CHANGE DISCOVERY

Figure 5.17: Deleting an edge from P 's instance graph for forming Q 's instance graph. Figure 5.18: Adding a new vertex into Q 's instance graph whose corresponding instance is not generated from P 's instance.

share an edge, its ancestors P_{v_1} and P_{v_2} do not share an edge either. It conflicts with the truth that P_{v_1} and P_{v_2} share edges in the input graph. The same conclusion can be obtained for the second operation of adding new vertices into the instance graph. Therefore, the instance graph of Q can only be built from that of P with the above three types of operations.

Theorem 5.1 *An instance graph based support measure for mining frequent/important subgraph in a single labeled graph has the downward closure property if its value is non-increasing under the forming clique, removing vertex, and adding edge operations happening in the instance graph.*

Proof: As demonstrated above, given a graph pattern P and its child Q , Q 's instance graph can be and only be constructed under the three types of operations on that of P . If an objective measure f is non-increasing under the three type of operations, the

Figure 5.19: An example showing that the G-Measure does not have the downward closure property.

value of P computed by f will not be lower than that of Q , $f(P) \geq f(Q)$, satisfying the definition of the downward closure property. $\square$

5.3.5 An Approximate Method of Computing G-Measure

It is expected that the proposed G-Measure has the downward closure property for mining G-Patterns efficiently. However, it is shown that such an expectation fails when applying the above three types of operations on the instance graph. For example, Figure 5.19 presents an example that Q 's instance graph is built by removing vertexes from P 's instance graph. Following the definition of G-Measure, Q 's value ($G(Q) = 2$) will be greater than that of P ($G(P) = 1$). As a result, the standard G-Measure does not have the downward closure property.

To tackle this problem, we utilize an approximate method of computing G-Measure for finding G-Patterns. The basic idea is based on the fact that all the instances of Q must be generated from those of its ancestor P . If we have partitioned P 's instance graph, we may not need to partition that of Q again. The vertices of Q 's instance graph can be firstly placed into different partitions according to its ancestors, and then are merged if necessary. By doing such, we can also obtain the G-Measure value of Q .

An example for illustrating this approach is given Figure 5.20, where Figure 5.20a is the instance graph of Q that we will process and Figure 5.20b is the three partitions of Q obtained based on P 's results. To get $G(Q)$, the exact way is to partition the instance

Figure 5.20: The two approaches of computing G-Measure value.

graph directly. However, we can also compute $G(Q)$ by merging the neighbor partitions 1 and 3, since merging them increases the modularity value.

The worst case of the approximate method is that some qualified graph patterns, e.g., the one shown in Figure 5.19, would be pruned. However, it can make all the incorrect patterns are removed. More importantly, the approximate method can have the downward closure property with the following lemmas and Theorem 5.1.

Lemma 5.1 *Given the instance graph of a graph pattern P and that of its child Q , if Q 's G-Measure is computed with the approximate method, the G-Measure of P will be no less than that of Q , $G(P) \geq G(Q)$, when forming clique operation happens on the instance graph.*

Proof: Given a vertex of P 's instance graph, v_p , that has been placed into a partition Par_i , and a clique K of Q that is generated from v_p ⁸,

- (i) if Par_i is not merged with other partitions, all the vertices of K will be placed in Par_i . No new partition is added on the instance graph and no existing partition is deleted, i.e., $G(P) = G(Q)$.

⁸For simplicity, we assume the three types of operations happen in the instance graph after the neighbor partitions have been merged.

- (ii) if Par_i is merged with other partitions to form a new partition, all the vertices of K will be placed in the new partition. The total number of partitions is reduced, i.e., $G(P) > G(Q)$. $\square$

Lemma 5.2 *Given the instance graph of a graph pattern P and that of its child Q , if Q 's G -Measure is computed with the approximate method, the G -Measure of P will be no less than that of Q , $G(P) \geq G(Q)$, when adding edge operation is conducted on the instance graph.*

Proof: Given two vertices of P 's instance graph, v_{pi} and v_{pj} , and two vertices of Q 's instance graph, v_{qi} and v_{qj} , where v_{qi} is generated from v_{pi} , v_{qj} is generated from v_{pj} , and an edge is added between v_{qi} and v_{qj} ,

- if v_{pi} and v_{pj} both are in a partition Par_i and Par_i is not merged with other partitions, v_{qi} and v_{qj} will also be placed in Par_i . No new partition is added on the instance graph and no existing partition is deleted, i.e., $G(P) = G(Q)$.
- if v_{pi} and v_{pj} both are in a partition Par_i and Par_i will be merged with other partitions, v_{qi} and v_{qj} both will be in the new partition. The total number of partitions is reduced, i.e., $G(P) > G(Q)$.
- if v_{pi} is in partition Par_i , v_{pj} is in partition Par_j , and Par_i will not be merged with Par_j , v_{qi} is in Par_i and v_{qj} is in Par_j subsequently. No new partition is added and no existing partition is deleted, i.e., $G(P) = G(Q)$.
- if v_{pi} is in partition Par_i , v_{pj} is in partition Par_j , and Par_i will be merged with Par_j , v_{qi} and v_{qj} will be placed in the same new partition. The total number of partitions is reduced, i.e., $G(P) > G(Q)$. $\square$

Lemma 5.3 *Given the instance graph of a graph pattern P and that of its child Q , if Q 's G-Measure is computed with the approximate method, the G-Measure of P will be no less than that of Q , $G(P) \geq G(Q)$, when removing vertex operation is conducted on the instance graph.*

Proof: Given a vertex of P 's instance graph, v_p , that is placed in a partition Par_i and v_p should be removed from the instance graph of Q ,

- If Par_i has more vertices besides v_p and Par_i is not merged with other partitions, Par_i will still exist after removing v_p , i.e., $G(P) = G(Q)$.
- If Par_i has more vertices besides v_p and Par_i will be merged with other partitions, the total number of partitions is reduced, i.e., $G(P) > G(Q)$.
- If Par_i only contains v_p , removing v_p will also lead to deleting Par_i . The total number of partitions is reduced, i.e., $G(P) > G(Q)$. $\square$

Another advantage of using this approximate method is that it can accelerate mining speed. In particular, we can treat the process of merging neighbor partitions as a function of looking for optimized G-Measure values. Two neighbor partitions are simply merged as if the modularity value increases, which is a local optimization of the G-Measure value. A global optimization has to analyze all the partitions of P for maximizing the modularity value. Because the local optimized value is always not less than the exact G-Measure value, we can quickly prune many spurious patterns by merging a few pairs of partitions.

5.4 G-Miner

In this section, we present the G-Miner algorithm designed with the approximate method of computing G-Measure values for mining G-Patterns. Specifically, we first introduce the Depth-First Search (DFS) code based approach [YH02] for candidate generation and graph isomorphism testing. Then, we present the details of the G-Miner algorithm.

Figure 5.21: An illustration of the DFS code based approach.

5.4.1 The DFS code

Graph isomorphism testing, subgraph isomorphism testing, and candidate generation are costly steps in the frequent subgraph mining problems. In G-Miner, we use the DFS code based approach to solve these problems.

The basic ideas behind the DFS code based approach are given as follows. For a labeled graph G , for example, the one shown in Figure 5.21a, we have different DFS trees of this graph (see T1 in Figure 5.21b and T2 in Figure 5.21c). We call edges included in the DFS tree are the forward edges and edges not in the DFS tree are the backward edges. A linear order $\prec$ of the edges in G is then defined. Particularly, given a DFS tree T and two edges $a = (vi_a, vj_a)$ and $b = (vi_b, vj_b)$ in G where vi_a is the start node of a and vj_a is the end node of a according to T ,

$$a < b = \begin{cases} a \text{ and } b \text{ are forward edges, } (vj_a < vj_b) \parallel (vi_a > vi_b \&\& vj_a = vj_b), \\ a \text{ and } b \text{ are backward edges, } (vi_a < vi_b) \parallel (vi_a = vi_b \&\& vj_a < vj_b), \\ a \text{ is a forward edge and } b \text{ is a backward edge, } vj_a \leq vi_b, \\ a \text{ is a backward edge and } b \text{ is a forward edge, } vi_a < vj_b. \end{cases} \quad (\text{Eq. 5.4})$$

With the $\prec$, the edges in G can be arranged into a sequence for individual DFS trees, called DFS code of G . For example, the DFS code for T1 is $\langle v0, v1, B, y, A \rangle$

, $\langle v1, v2, A, x, A \rangle$, $\langle v2, v3, A, y, C \rangle$, $\langle v3, v1, C, x, A \rangle$) and that of T2 is ($\langle v0, v1, A, x, A \rangle$, $\langle v1, v2, A, x, C \rangle$, $\langle v2, v0, C, y, A \rangle$, $\langle v1, v3, A, y, B \rangle$).

As a linear order over the labels in the graph can further be defined, for example, $A < B < C$ for the input graph G^9 , we are able to sort these different DFS codes obtained. For example, as $\langle v0, v1, A, x, A \rangle$ is smaller than $\langle v0, v1, B, y, A \rangle$ with the linear order over the labels, the DFS code of T2 is thus smaller than that of T1, which can also be proved to be the minimum DFS code of G.

Because two graphs are isomorphic if they have the same minimum DFS code, the problem of mining frequent subgraphs can be changed to mining frequent minimum DFS codes. Particularly, the edge sequence for generating a subgraph must be the minimum DFS code of this subgraph. As a result, only a few candidates would be explored, which reduces the total cost of subgraph isomorphism testing and candidate generation.

5.4.2 Algorithm Details

G-Miner utilizes a Depth-First Search method to find G-Patterns, since this approach has shown to have a computational advantage over the Breadth-First Search for mining a single labeled graph [KK05] and works efficiently with the DFS code. Specifically, G-Miner starts with a set of frequent 2-edge graph patterns. Then, it recursively generates potential descendants with a growth of one edge. Graph patterns with G-Measure value not less than the minimum support threshold are selected.

Figure 5.22 outlines the pseudo-code of the G-Miner algorithm. It is similar to the original **gSpan** algorithm [YH02]. The major differences between the two algorithms are that G-Miner will partition the instance graphs and use the G-Measure for finding frequent subgraphs. Explanations of this algorithm are presented as follows.

⁹Note that we can order the labels using other approaches besides the lexicographic order. For example, in [YH02], the labels are ordered according to their frequencies in the data set.

G-Miner Algorithm

Input: G : a labeled graph.
 θ : a user-specified minimum support threshold.
Output: S : frequent subgraph set.

main(G, θ, S) // the main function

1. sort the labels of the vertices and edges in G by their frequency
2. remove infrequent edges and vertices
3. relabel the remaining vertices and edges in descending frequency
4. sort the 1-edge graphs into S^1 referring to the DFS lexicographic order
5. **foreach** 1-edge graph $sg \in S^1$
6. enumerate sg 's potential 2-edge children
7. **foreach** 2-edge graph P , P is a child of sg
8. record P 's instances in G
9. build and partition P 's instance graph
10. assign each instance of P into its corresponding partition
11. **subgraph_mining**(G, S, P)
12. $S \leftarrow S \cup S^1$

subgraph_mining(G, S, P) //the sub procedure

13. merge different partitions into a new partition if possible
14. compute the G-Measure value of P , $G(P)$
15. **if** $G(P) < \theta$ **then**
16. **return** NULL
17. $S \leftarrow S \cup \{P\}$
18. generate all P 's potential children with one edge growth
19. *// only patterns with minimum DFS code are generated*
19. **foreach** P^* , P^* is a child of P
20. record the instances of P^* in G
21. assign each instance into a particular partition according to its parent
22. **subgraph_mining**(G, S, P^*)

Figure 5.22: The Pseudo-Code of the G-Miner Algorithm.

Step 1 (line 1-4): Scan the whole input graph G into the memory. Remove infrequent vertices and edges from the input graph. The remaining vertices and edges are relabeled in descending frequency for forming 1-edge graph patterns. These frequent 1-edge graphs are added into S^1 and sorted in the DFS lexicographic order. Note that in this paper we only consider that the whole data set can fit into main memory. The problem of processing large scale data sets that cannot be held into main memory will be our future work.

Steps 2-4: Loops to generate qualified G-Patterns. It stops after all the elements of S^1 are explored.

Step 2 (line 6): For each 1-edge graph pattern, enumerate its potential 2-edge children. Note that only graph patterns satisfying the minimum DFS code are built.

Step 3 (line 8-10): For each 2-edge graph pattern P , record P 's instances in the input graph G , build P 's instance graph and partition this graph. Each instance will be assigned into a particular partition according to the partition result.

Step 4 (line 11): Subgraph_mining recursively generates P 's potential descendants with a growth of one edge (satisfying the minimum DFS code). Only graph patterns with G-Measure value greater than the minimum support are added to S (**line 17**) and further explored (**line 18**). Their instances are firstly placed into different partitions according to the ancestors (**line 21**). Then, neighbor partitions are merged as new partitions (similar to the agglomerative clustering) if the modularity value increases (**line 13**). For patterns whose G-Measure values are already lower than the threshold before finishing the merge process, they are discarded immediately.

Step 5 (line 12): Add the frequent 1-edge graph set S^1 to S .

Note that the reason why G-Miner starts from 2-edge patterns but not 1-edge patterns is because the instance graphs of the 1-edge patterns are all composed of a set of disconnect vertices. As a result, the G-Miner algorithm cannot separate instances

Table 5.1: Parameter settings for the synthetic data used in the experiments.

Data	$ V $	$ E $	N	L	α
V2E1N500L20A55	2	1	500	20	0.55
V2E1N500L20A05	2	1	500	20	0.05
V2E1N550L10A65	2	1	550	10	0.65
V2E1N550L100A65	2	1	550	100	0.65

that are located in the same area from those appearing in different areas, making poor performance for finding G-Patterns.

5.5 Experiments

In this section, we evaluate the performance of the G-Miner algorithm for mining G-Patterns. Specifically, we evaluate: (1) the effectiveness of the G-Miner algorithm for mining G-Patterns. (2) the computational efficiency of the G-Miner algorithm for mining G-Patterns.

5.5.1 Experimental Setup

We use both synthetic and real-world data in the experiments for evaluating G-Miner’s performance.

Synthetic Data. We develop a synthetic data generator which is based on Barabasi’s evolving scale-free random graph model [BA99] to produce labeled power-law graphs in the experiments, as we can easily observe the difference between the patterns mined by the standard frequent subgraph mining algorithms and those discovered by G-Miner on such graphs.

The procedure of this generator for producing labeled graphs is as follows: Firstly, by setting the initial vertex seed number $|V|$, the number of edges to be added at each step $|E|$, and the number of steps for evolution N , an unlabeled power-law graph is generated. Then, each vertex is assigned a particular label, which is controlled by the number of

Table 5.2: The characteristics of the real-world data.

Data	Total number of		Total number of	
	Vertices	Edges	Vertex label	Edge label
credit	14,700	14,000	59	20
aviation	101,185	98,576	6,173	51
citation50_25	18,114	47,776	50	1
citation50_20	15,766	34,494	50	1
citation50_15	12,760	21,219	50	1
PPI	4,931	10,003	28	1

labels in the graph (L) and the probability of neighbor vertices sharing a same label (α). The details how the parameters L and α affect the patterns mined are given in Section 5.5.2. The summary of the parameter settings for creating the synthetic data sets is given in Table 5.1.

Real-world Data. Six real-world data are used in the experiments. The basic characteristics of these data are shown in Table 5.2.

The *credit* and *aviation* data are downloaded from SUBDUE’s web site¹⁰. Among all, the *aviation* data is the largest graph used in the experiments, which consists of more than 100,000 vertices and 90,000 edges.

The *citation50_15*, *citation50_20*, and *citation50_25* data are generated from the citation graph used in the KDD Cup 2003¹¹, the vertices of which represent research papers and the edges of which indicate the citation relation between papers. Because the original citation graph does not have any meaningful labels for the vertices and edges, we firstly apply a clustering algorithm to group the abstracts of the papers into 50 topic clusters. Then, the cluster IDs are assigned to the vertices as their labels. For the edges, we give them a common label, since they all represent the citation relation. As our experiments are assumed for processing undirected graphs only, the directions of the edges are ignored. Note that the original citation graph is very dense, containing 29,555 vertices

¹⁰<http://www.subdue.org>

¹¹<http://www.cs.cornell.edu/projects/kddcup/>

Figure 5.23: A 4-edge pattern with frequency of 1,843,068 in the original citation graph.

and 352,807 edges. A simple graph pattern can have even many instances in the data (for example, the one shown in the Figure 5.23 even has a frequency of 1,843,068), which leads to the programs quickly run out of memory even just starting mining. Therefore, we only use part of the original graph for evaluation, which is created by removing vertices with degree greater than a threshold θ and their associated edges from the original graph. In our experiments, we set the threshold as 15, 20, and 25, respectively¹². Such settings enable us to conduct a more comprehensive study of the edge density to the G-Pattern mined and allow the computation to be finished in a reasonable amount of time.

The *PPI* data is a protein-protein interaction network¹³, in which each vertex represents a protein and an edge is added between two vertices if the corresponding proteins are detected to have an interaction in the experiments. For each vertex, we assign the protein's functional class as its label¹⁴. Since all the edges represent the interaction relation, we only give one label to them. Similar to the citation graphs, we remove vertices whose degrees are greater than 10 and the incident edges from the original graph to guarantee the computation to be done in a reasonable amount of time.

Baseline Algorithms. In our experiments, we implement two algorithms, namely

¹²When setting a higher threshold θ , the resulted graph would be denser, resulting in more subgraphs to be extracted with more time.

¹³The PPI data is available at <http://dip.doe-mbi.ucla.edu/dip/Download.cgi>

¹⁴The functional class scheme used is the one established by the MIPS scheme, which is available in <http://mips.gsf.de/projects/funccat>

Baseline-1 AlgorithmInput: G : a labeled graph. θ : a user-specified minimum support threshold.Output: S : frequent subgraph set.**main**(G, θ, S) // the main function

1. sort the labels of the vertices and edges in G by their frequencies
2. relabel the remaining vertices and edges in descending frequency
3. sort the 1-edge graphs into S^1 according to the DFS lexicographic order
4. **foreach** 1-edge graph $sg \in S^1$
5. **subgraph_mining**(G, S, sg)
6. $S \leftarrow S \cup S^1$

subgraph_mining(G, S, P) //the sub procedure

7. compute the support value of P , $S(P)$
8. **if** $S(P) \geq \theta$ **then**
9. $S \leftarrow S \cup \{P\}$
 // Cannot stop mining here because of missing downward closure property.
10. generate all P 's potential children with one edge growth
11. **foreach** P^* , P^* is a child of P
12. **subgraph_mining**(G, S, P^*)

Figure 5.24: The pseudo-code of the baseline-1 algorithm.

baseline-1 and baseline-2, as the baselines for comparison.¹⁵ The details of the two algorithms are given as follows.

The baseline-1 algorithm is a standard algorithm for finding frequent subgraphs in a single labeled graph, in which the support value of a graph pattern is computed as the exact number of its instances in the input graph. Its pseudo-code is given in Figure 5.24. The standard G-Measure can be directly incorporated into this algorithm, either during the mining process or as a post-process step, for mining G-Patterns.

As mentioned previously, the standard support measure may not have the downward closure property for mining a single labeled graph. To overcome this problem, a downward

¹⁵We do not compare with the *divide and conquer* approach in this thesis as we do not find good methods to set the k value and to recover the lost patterns.

closure property holding measure is proposed by Vanetik *et al.* [VGS02] that assigns the size of the maximum independent set (MIS) of a pattern's instance graph as its support value. Algorithm with this support measure are supposed to quickly discover a set of frequent subgraphs from the input graph, although it cannot find all the frequent subgraphs. Here, we further implement the baseline-2 algorithm with Vanetik's support measure for better evaluating the computational performance of the G-Miner algorithm. Its pseudo-code is given in Figure 5.25.

Baseline-2 Algorithm

Input: G : a labeled graph.

θ : a user-specified minimum support threshold.

Output: S : frequent subgraph set.

main(G, θ, S) // the main function

1. sort the labels of the vertices and edges in G by their frequencies
2. remove infrequent edges and vertices
//baseline-1 cannot conduct this operation
3. relabel the remaining vertices and edges in descending frequency
4. sort the 1-edge graphs into S^1 according to the DFS lexicographic order
5. **foreach** 1-edge graph $sg \in S^1$
6. **subgraph_mining**(G, S, sg)
7. $S \leftarrow S \cup S^1$

subgraph_mining(G, S, P) //the sub procedure

8. compute maximum independent set of P 's instance graph, $MIS(P)$
9. **if** $MIS(P) < \theta$ **then**
10. **return** NULL
11. $S \leftarrow S \cup \{P\}$
12. generate all P 's potential children with one edge growth
13. **foreach** P^* , P^* is a child of P
14. **subgraph_mining**(G, S, P^*)

Figure 5.25: The pseudo-code of the baseline-2 algorithm.

All the experiments are conducted on an Intel 64-bit Xeon CPU (2.0 GHz) PC with 4G main memory, running Linux. When partitioning the instance graph, we use a fast

community detection algorithm [CNM04] with time complexity of $O(md \log n)$, where n is the number of vertices, m is the number of edges, and d is the depth of the dendrogram. When computing the MIS of the instance graph, we use a fast exact maximum clique algorithm *wclique* [Ö02]. Note that the reason of using a 64-bit PC is because the 32-bit JVM can only support up to 1.5 Gigabyte memory while some data used require more memory. In our experiments, the maximum memory heap size is 3G. Also, as our program is implemented using JAVA, it gives us a disadvantage when comparing with programs using C/C++.

5.5.2 Effectiveness for Mining G-Patterns

Firstly, we evaluate the effectiveness of the G-Miner algorithm for mining G-Patterns. As the real-world data are too large for visualization, we only use the synthetic data here. For graphs V2E1N500L20A55 and V2E1N500L20A05, we set a lower α value to the latter so that the probability of two linked vertices sharing a common label is more random, which leads to fewer frequent patterns to be found. For graphs V2E1N550L10A65 and V2E1N550L100A65, assigning more labels to the latter also leads to fewer frequent patterns to be found.

The experimental results of the G-Miner algorithm on the synthetic data are given in Figure 5.26, Figure 5.27, Figure 5.28, and Figure 5.29, respectively. We have highlighted vertices which are contained in the top ten frequent multi-edge G-Patterns found in these data. Also, for comparison, we mark the coverage of the top ten frequent multi-edge patterns found by baseline-1 whose edge number is not greater than the maximal edge number of the patterns discovered by G-Miner¹⁶.

We can see the frequent subgraph patterns found by baseline-1, as we expect, occur only in several particular areas of the input graphs (particularly, these found graph pat-

¹⁶We do not show the coverage of the patterns found by the standard G-Measure in the figures, as they are the same as those found by G-Miner.

CHAPTER 5. MINING ONTOLOGY INSTANCE DATA FOR CHANGE DISCOVERY

Figure 5.26: The coverage of the top ten frequent multi-edge patterns extracted by G-Miner compared with that of baseline-1 on graph V2E1N500L20A55.

Figure 5.27: The coverage of the top ten frequent multi-edge patterns extracted by G-Miner compared with that of baseline-1 on graph V2E1N500L20A05.

CHAPTER 5. MINING ONTOLOGY INSTANCE DATA FOR CHANGE DISCOVERY

Figure 5.28: The coverage of the top ten frequent multi-edge patterns extracted by G-Miner compared with that of baseline-1 on graph V2E1N550L10A65.

Figure 5.29: The coverage of the top ten frequent multi-edge patterns extracted by G-Miner compared with that of baseline-1 on graph V2E1N550L100A65.

Table 5.3: The number of unique vertices in the patterns found in the synthetic data.

Data	G-Miner	Baseline-1
V2E1N500L20A55	107	52
V2E1N500L20A05	59	20
V2E1N550L10A65	185	54
V2E1N550L100A65	44	22

terns all occur around the vertices with high degrees), while patterns found by G-Miner distribute globally in the synthetic graphs. The results clearly demonstrate the effectiveness of the G-Miner algorithm for mining G-Patterns. Also, Table 5.3 lists the number of unique vertices found by the two algorithms. Although the number of frequent subgraphs to be mined from each synthetic graph is different, G-Miner can always effectively find G-Patterns which cover more unique vertices of the input graphs.

5.5.3 Computational Efficiency of Mining G-Patterns

We evaluate the computational performance of the G-Miner algorithm on the six real-world data. The baseline-1 and baseline-2 algorithms are used in the experiments for comparison, since the standard G-Measure can be incorporated into them, either during the mining process or as a post-process step, for finding G-Patterns. If G-Miner outperforms the two algorithms in the experiments, it is proved to be computationally efficient for mining G-Patterns.

Table 5.4 shows the runtime (in seconds), the minimum support threshold used, and the number of frequent patterns found by the three algorithms. Entries marked with “-” represent experiments aborted for running more than one day or out of memory. We can see G-Miner runs greatly faster than baseline-1 and baseline-2 on these data sets. Given the same minimum support threshold and 24-hour slot, baseline-1 cannot work on any data sets and baseline-2 works only on three. Such results clearly demonstrate the computational efficiency of the G-Miner algorithm for finding G-Patterns.

Table 5.4: The experimental results of G-Miner, baseline-1, and baseline-2 on the six real-world data.

data	θ	G-Miner		Baseline-1		Baseline-2	
		runtime	#patterns	runtime	#patterns	runtime	#patterns
credit	50	155.2	73,992	-	-	432.6	73,992
	100	33.4	11,696	-	-	217.9	11,696
aviation	1,500	80.7	5,231	-	-	88,623	5,231
	2,000	16.6	843	-	-	24,988	843
citation50_15	10	2.1	1,099	-	-	2.9	1,140
	20	1.1	260	-	-	1.1	260
citation50_20	30	-	-	-	-	-	-
	40	1.45	108	-	-	-	-
citation50_25	70	1.57	65	-	-	-	-
	80	1.34	53	-	-	-	-
PPI	250	-	-	-	-	-	-
	275	286.5	12	-	-	-	-

5.6 Analysis of China's Stock Market: A Case Study

For an overall evaluation of the G-Pattern based approach to change discovery, we apply the proposed G-Miner algorithm on a real-world data set (CSM) that collects the relevant information, e.g., share holding and product transaction, of the listed companies in China's stock market from 2002 to 2005. An example of the ontology and data in the data set is given Figure 5.30. Although the concepts (e.g., *Fund Company*) do not change, the major relations between the concepts (e.g., *share holding*) evolve over time. We aim to find the major relations between these concepts over a period of time with the G-Patterns, which will help to update the ontology, and monitor the anomalous actions in the stock market.

Two graphs are built from the data set for the experiment. The first one (CSM2002) collects information from 2002/01/01 to 2002/03/31, which consists of 23,921 instances with 23,581 relations between these instances. The second one (CSM2005) collects information from 2005/01/01 to 2005/03/31, which is composed of 24,160 instances and

CHAPTER 5. MINING ONTOLOGY INSTANCE DATA FOR CHANGE DISCOVERY

Figure 5.30: An example of the CSM data set.

Figure 5.31: The number of G-Patterns mined from the graphs CSM2002 and CSM2005 with different minimum support value and the runtime.

CHAPTER 5. MINING ONTOLOGY INSTANCE DATA FOR CHANGE DISCOVERY

Figure 5.32: Sample patterns mined from the graphs CSM2002 with minsup 100.

23,188 relations. Given China's impressive economic development during these years, the G-Patterns mined from the two graphs are expected to be different.

Firstly, we check the possible G-Patterns to be mined from the two graphs with different minimum support values (*minsup*). The number of G-Patterns mined and the runtime with a *minsup* of 300, 250, 200, 150, 100, and 50, are given in Figure 5.31. We can see more G-Patterns are mined from the graph CSM2005, which incurs more time.

Then, we analyze the patterns mined from the two graphs with a *minsup* of 100.

For graph CSM2002, there is only 56 G-Patterns mined, 37 of which are the single edge patterns (an example is given in Figure 5.32a, which is about the concept *Listed Automobile Company*.), and 22 of which about the concept *Listed Pharmaceutical Company*. The max edge length G-Pattern found is given in Figure 5.32b, which contains three concepts and two relations.

For graph CSM2005, there is a total of 144 G-Patterns discovered, most of which are about the concepts *Listed Pharmaceutical Company*, *Listed Energy Company*, *Listed Automobile Company*, and *Listed Agricultural Company*, and only 42 of which are single edge patterns. The max edge length G-Pattern of the concept *Listed Automobile Company* is given in Figure 5.33a, which involves five concepts and three relations. The max

CHAPTER 5. MINING ONTOLOGY INSTANCE DATA FOR CHANGE DISCOVERY

Figure 5.33: Sample patterns mined from the graphs CSM2005 with minsup 100.

edge length G-Pattern mined in this graph is given in Figure 5.33b, which includes four concepts and four relations.

We can see the patterns found in the two graphs are greatly different, which verifies our hypothesis on the patterns to be mined. Furthermore, an interesting thing is observed that the G-Patterns mined can predict the trend of the stock market. For example, for CSM2002, it is in the middle of the four-year slump of China's stock market, where the max edge length G-Pattern extracted contains only a few concepts and relations. In 2005, a ban on new IPOs was put by China government to curb the slump. Corresponding, the max edge length G-Pattern mined in CSM2005 involves more concepts and relations. The detailed relationship between the G-Patterns and the trend of the stock market would be studied in the future.

5.7 Related Work

In recent years, there are a large number of data mining algorithms developed for analyzing and managing graph structured data. Based on the difference of the problem formulation, we can classify the existing graph mining algorithms into two categories, namely mining frequent/important patterns that appear across a set of small graphs and mining frequent/important patterns that occur in a single graph. The G-Miner algorithm presented in this chapter belongs to the second category.

Between the two classes of algorithms, algorithms belonging to the first category [IWM00, KK01, YH02, WWP⁺04, WHLS06, KKS⁺06] are more mature. These developed algorithms can efficiently and effectively discover all the frequent subgraphs in the data and are scaled to large data sets. In contrast, algorithms of the second category have received much less attention. Besides the G-Miner algorithm, there are only a few attempts for finding frequent/important subgraphs in a single labeled graph [HCD94, YM95, VGS02, KK05, GC02]. As a major difference, the G-Miner is aimed at finding globally distributed frequent subgraphs in the input graph. In addition to this key difference, we compare and contrast some algorithmic ideas of the related mining algorithms as follows.

The SUBDUE system [HCD94] is a well-known algorithm which selects qualified patterns for compressing the input graph under the minimum description length principle. Specifically, SUBDUE searches for patterns P s in the input graph G that minimize

$$I(P) + I(G|P)$$

where $I(P)$ represents the number of bits required to encode P and $I(G|P)$ is the number of bits required to encode G with respect to P . This requirement makes the patterns found by SUBDUE may not be the most frequent patterns in the graph. Furthermore, to avoid the problem of exploring the whole search space, a heuristic beam search is

employed to narrow the search space. Only limited candidates are kept for further exploration. As a result, SUBDUE cannot find the complete set of important/frequent subgraphs in the input graph. The similar approach is adopted by GBI [YM95] for finding typical subgraphs in a single labeled graph.

The SEuS algorithm [GC02] aims at finding frequent subgraphs in a directed labeled graph. In particular, SEuS first uses a data structure called *data summary* to construct a compressed representation of the input graph, similar to DataGuides [GW97] for semi-structured data. Then, the frequencies of the graph patterns are estimated based on the built *data summary*. As such, SEuS can quickly find the most frequent patterns in the input graph. However, it still needs to explore the whole search space for finding all the qualified frequent patterns. As the authors indicate [GC02], SEuS is not efficient if the input graph contains a large number of frequent subgraphs with low frequency and is useful only when there are a few frequent subgraphs with high frequency in the input graph. This limitation greatly affects its extension of mining G-Patterns, since G-Patterns are normally with relatively low frequencies.

Given the fact that the standard support measure does not guarantee the downward closure property for finding frequent subgraphs in a single labeled graph, Vanetik et al. [VGS02] propose the downward closure property holding support measure which assigns the size of the MIS of a pattern's instance graph as the support value. Because of the downward closure property, algorithms implemented with this support measure may quickly find a set of frequent subgraphs in a single labeled graph. However, computing the maximum independent set of a graph has proved to be a NP-hard problem [GJ79], which can make the problem even worse. Also, as shown in [KK05], algorithms implemented with this measure are only efficient for mining large sparse graphs due to the high cost of computing the maximum independent set. As a result, the new support measure cannot solve the problem fundamentally.

5.8 Summary

The contributions of this chapter are summarized as follows:

- Different from the existing methods, we view the ontology instance data as a single labeled graph and exploit the frequent subgraph mining technique for change discovery. The advantage of adopting such an approach is that the frequent subgraphs discovered can not only indicate which items are closely related but also how they are related, which are greatly helpful for change discovery.
- The *Mining Globally Distributed Frequent Subgraphs in a Single Labeled Graph* problem is identified when adopting the frequent subgraph mining technique for change discovery. These globally distributed frequent subgraphs discovered, i.e., *G-Patterns*, can provide global information about the entire graph and are valuable for a variety of applications such as web usage mining, hypertext classification, and biological data analysis.
- To effectively mine G-Patterns, an objective measure called *G-Measure* is developed, which is to partition the instance graphs of the graph patterns for identifying G-Patterns and can be directly incorporated into the existing frequent subgraph mining algorithms. Furthermore, in view that the standard G-Measure does not have the downward closure property, an approximate method of computing G-Measure is designed which can automatically remove all spurious patterns during the mining process and incorporated into a *G-Miner* algorithm for mining G-Patterns. The experimental results on both synthetic and real-world data show that the G-Miner algorithm can effectively identify G-Patterns and run much faster than the alternative ways of mining G-Patterns.

Chapter 6

Conclusions and Future Work

This thesis has presented the methods and systems developed for facilitating the use of the ontology based solutions for information service. Specifically, we have proposed, implemented, and evaluated a suite of techniques and algorithms spanning ontology learning from domain specific text documents, ontology based user modelling, ontology based information retrieval, and ontology instance data driven change discovery.

This chapter is aimed at retrospecting the challenges we have identified when adopting the ontology based solution for providing information services and summarizing the methods and systems developed to tackle the problems.

6.1 Contributions

6.1.1 Ontology Building

Ontology learning is aimed at building an ontology from scratch, or adapting an existing ontology in an automatic or semi-automatic fashion from relevant resources so as to ease the effort of building an ontology manually. In view of the problems of the existing ontology learning systems, we develop a system, called Concept-Relation-Concept Tuple based Ontology Learning (CRCTOL), for mining ontologies automatically from domain specific text documents.

CHAPTER 6. CONCLUSIONS AND FUTURE WORK

Table 6.1: Summary of key features compared with other ontology learning systems.

	OntoBuilder	OntoLearn	Text-To-Onto /Text2Onto	CFG	CRCTOL
NLP tools	Not Known	POS tagger Chunk parser	POS tagger	POS tagger	POS tagger Full parser
Concept Extraction Measure	Not Known	DR-DC	tf/idf	No	DRM
Taxonomic Relation Extraction	Yes	Yes	Yes	No	Yes
Non-Taxonomic Relation Extraction	No	Not Known	Yes	Yes	Yes

In particular, the developed CRCTOL system differs from other state-of-the-art ontology learning systems in a number of ways (Table 6.1). First, we adopt a full text parsing technique to obtain more detailed syntactic level information. Second, we use a different procedure including the developed *DRM* measure for concept extraction, which enables us to extract a concise set of domain specific concepts more accurately. Third, we design a simple but effective unsupervised word sense disambiguation algorithm to identify the intended meaning of each word, which helps build the correct relations between concepts. Finally, we use a rule-based method similar to the CFG approach for non-taxonomic relation extraction. This proves to be a feasible way to extract previously unknown semantic relations. As shown in the experimental results, our system produces ontologies that are more concise and accurate, and contain a richer semantics in terms of the range and number of semantic relations extracted compared with alternative systems.

6.1.2 Ontology Exploitation

One major goal of adopting ontology based solutions is to effectively and knowledgeably address each individual's information requirements. In this thesis, the goal is achieved by developing an ontology based user model, called *user ontology*, for supporting personalized services. Specifically, a user ontology is a specialization of a domain ontology

by assigning each concept and relation of the domain ontology with a specific value for indicating a user's interests. It is a personalized view of the domain conceptualization and is more comprehensive than the existing types of user models in representing a user's interests in a particular domain.

We have developed a set of methods for learning and inferencing in the user ontology model and integrated it into a semantic search engine called *OntoSearch* for performing personalized document retrieval. It is a seamless extension of the *OntoSearch* system with the added advantage of using user ontology. The experimental results, on the ACM digital library and the Google Directory document sets, support the efficacy of the user ontology model and the validity of learning and exploiting user ontology.

6.1.3 Ontology Analysis

For enabling up-to-date services, ontologies have to evolve over time. To support a continual evolution of ontologies, people have developed various methods for capturing the necessary changes in the ontologies. In this thesis, we employ the ontology instance data driven approach for tackling this problem.

In particular, different from the existing methods, we view the ontology instance data as a single labelled graph and apply the frequent subgraph mining techniques for change discovery. To this end, we identify a new problem of *Mining Globally Distributed Frequent Subgraphs in a Single Labelled Graph*. The patterns introduced by this problem, called *G-Patterns*, provide global and balanced information about the entire graph and are valuable for a variety of applications.

To effectively mine G-Patterns, an objective measure called *G-Measure* is firstly developed, which is to partition the instance graphs of the graph patterns for identifying G-Patterns and can be directly incorporated into the existing mining algorithms. Then, in view that the standard G-Measure does not guarantee the downward closure property,

an approximate method of computing the G-Measure is designed which can automatically remove all the spurious patterns during the mining process and incorporated into a G-Miner algorithm for mining G-Patterns. The experimental results on both synthetic and real-world data show that the G-Miner algorithm can effectively mine G-Patterns and run much faster than the alternative algorithms.

6.2 Outstanding Issues

The following sections describe how the methods and systems presented in this thesis can be improved and refined.

6.2.1 Ontology Building

The CRCTOL system presented in this thesis is to learn ontologies automatically from document specific text documents. For enhancement, the CRCTOL system can be extended in two directions.

Firstly, the function of enriching an existing ontology and adapting an ontology for other application domains is not supported by the current CRCTOL system. The ontology has to be learned from scratch at each time, which is quite inefficient. For tackling this problem, people have proposed possible methods, such as POM model [CV05] that records the distributions of the concepts and relations in the corpus for reusing existing ontologies. However, all existing methods have their limitations that prevent them from offering a generic solution to this problem. Effective methods are therefore needed for supporting such a function in CRCTOL.

Secondly, in view of the problems suffered by the shallow NLP techniques for ontology learning, CRCTOL adopts a full text parsing technique instead, which leads to the learned ontology being more concise and containing a richer semantics. However, the shallow NLP techniques have their own merit for ontology learning. That is, they can run much faster

than the full text parsing technique. Also, in some cases, the shallow NLP techniques are actually sufficient for learning ontologies. Consequently, how to effectively combine the two NLP techniques for ontology learning becomes an interesting work, since it would greatly improve the efficiency of the CRCTOL system for learning ontologies from large document collections.

6.2.2 Ontology Exploitation

The proposed user ontology model can be extended in two aspects for providing better personalized services.

Firstly, as shown in the experiments, the performance of the user ontology model for providing personalized services would be affected by the size of the ontology used. To guarantee the effectiveness of the user ontology model for capturing users' interests, a longer training time is required for working with large ontologies. Therefore, we shall explore possible methods for facilitating this training process. For example, by referring to the profiles of the users with similar interests, a relatively precise user ontology can be constructed within a shorter time [SSM03].

Secondly, when inferencing in the user ontology, the spreading activation process can only stop after all the nodes of the network are activated with certain activation values, which would be quite time consuming for large networks. However, it is shown that only a few of concepts are relevant to the users' current interests and needed to be processed. The use of the whole ontology is in fact unnecessary for inferencing the users' interests. For improving the system's efficiency, methods that can compress the user ontology for the spreading activation process while still represent the users' interests as effectively as the use of the original ontology for inferencing are worthwhile to explore.

6.2.3 Ontology Analysis

For ontology instance data driven change discovery, the G-Pattern mining methods can also be extended in two directions. The first direction is to incorporate domain specific knowledge in the mining process for producing more informative patterns. For example, although no pattern of a concept A is founded in the ontology instance data, we can also find necessary changes about A if the patterns of its child concept B are mined. This is similar to the generalized association mining problem [SA95] but would be more complex, since there are other semantic relations in the ontology besides the taxonomic relations [HWAMAS04]. The other direction is to extend the G-Miner algorithm for handling large disk based graphs. Currently, the G-Miner algorithm assumes that the data are able to fit into the main memory, where the computation is only CPU-bounded. However, ontology instance data are normally too large to fit into the main memory. To process such data, the algorithm has to repeatedly scan and read the disk during the mining process, which makes the computation I/O intensive instead. For efficiently discovering changes in the ontologies, methods that support the scalable mining of large disk based graphs are worthy of investigation.

Appendix A

List of Publications

A.1 Journals

- (i) Xing Jiang and Ah-Hwee Tan, *Learning and inferencing in user ontology for personalized Semantic Web search*, Information Sciences, 179(16):2794-2808, 2009
- (ii) Xing Jiang, Hui Xiong, Chen Wang, and Ah-Hwee Tan, *Mining globally distributed frequent subgraphs in a single labeled graph*, Data & Knowledge Engineering (DKE), 68(10):1034-1058, 2009.
- (iii) Xing Jiang and Ah-Hwee Tan, *CRCTOL: A Semantic Based Domain Ontology Learning System*, Journal of the American Society for Information Science and Technology (JASIST), 61(1):150-168, 2010.

A.2 Conferences

- (i) Xing Jiang and Ah-Hwee Tan, *Mining Ontological Knowledge from Domain-Specific Text Documents*, In Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), pp. 665-668, Houston, Texas, USA, 27-30 November 2005.
- (ii) Xing Jiang and Ah-Hwee Tan, *Learning and inferencing in user ontology for personalized semantic web services*, In Proceedings of the 15th international conference

CHAPTER A. LIST OF PUBLICATIONS

- on World Wide Web (WWW 2006), pp. 1067-1068, Edinburgh, Scotland, UK, May 23-26, 2006.
- (iii) Xing Jiang and Ah-Hwee Tan, *OntoSearch: A Full-Text Search Engine for the Semantic Web*, In Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference (AAAI 2006), pp. 1325-1330, Boston, Massachusetts, USA, July 16-20, 2006.
- (iv) Huan Wang, Xing Jiang, Liang-Tien Chia, and Ah-Hwee Tan, *Ontology enhanced web image retrieval: aided by wikipedia & spreading activation theory*, In Proceedings of the 1st ACM SIGMM International Conference on Multimedia Information Retrieval (MIR 2008), pp. 195-201, Vancouver, British Columbia, Canada, October 30-31, 2008.

References

- [ABB⁺00] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.
- [And76] John R. Anderson. *Language, Memory and Thought*. Erlbaum, Hillsdale, N. J., 1976.
- [And83a] John R. Anderson. *The Architecture of Cognition*. Harvard University Press, Cambridge, MA, 1983.
- [And83b] John R. Anderson. A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behavior*, 22:261–295, 1983.
- [And93] John R. Anderson. *Rules of the Mind*. Erlbaum, Hillsdale, N. J., 1993.
- [AR96] Eneko Agirre and German Rigau. Word sense disambiguation using conceptual density. In *Proceedings of the 16th Conference on Computational Linguistics (COLING)*, pages 16–22, 1996.
- [ARK06] Amine Abou-Rjeili and George Karypis. Multilevel algorithms for partitioning power-law graphs. In *20th International Parallel and Distributed Processing Symposium (IPDPS)*, page 10, 2006.
- [AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of 20th International Conference on Very Large Data Bases (VLDB)*, pages 487–499, 1994.

REFERENCES

- [AW06] Ralitsa Angelova and Gerhard Weikum. Graph-based text classification: learn from your neighbors. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 485–492, 2006.
- [BA99] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.
- [BB04] Marco Baroni and Sabrina Bisi. Using cooccurrence statistics and the web to discover synonyms in technical language. In *The fourth international conference on Language Resources and Evaluation (LREC)*, pages 1725–1728, 2004.
- [BBT06] Marco Bertini, Alberto Del Bimbo, and Carlo Torniai. Automatic annotation and semantic retrieval of video sequences using multimedia ontologies. In *Proceedings of the 14th ACM International Conference on Multimedia (MM)*, pages 679–682, 2006.
- [BC05] Bo Brinkman and Moses Charikar. On the impossibility of dimension reduction in ℓ_1 . *J. ACM*, 52(5):766–788, 2005.
- [BdRP97] Roberto Basili, Gianluca de Rossi, and Maria Teresa Pazienza. Inducing terminology for lexical acquisition. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 125–133. 1997.
- [BF06] Tamara L. Berg and David A. Forsyth. Animals on the web. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1463–1470, 2006.
- [BLC96] Amaia Bernaras, Iñaki Laresgoiti, and Jose Manuel Corera. Building and reusing ontologies for electrical network applications. In *12th European Conference on Artificial Intelligence (ECAI)*, pages 298–302, 1996.
- [BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 285(5):34–43, 2001.

REFERENCES

- [BLK⁺09] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009.
- [BNC00] Gilles Bisson, Claire Nédellec, and Dolores Cañamero. Designing clustering methods for ontology building - the mo’k workbench. In *Proceedings of the Workshop on Ontology Learning, 14th European Conference on Artificial Intelligence*, 2000.
- [BOS04] Paul Buitelaar, Daniel Olejnik, and Michael Sintek. A protégé plug-in for ontology extraction from text based on linguistic analysis. In *The Semantic Web: Research and Applications, First European Semantic Web Symposium (ESWS)*, pages 31–44, 2004.
- [BP02] Satanjeev Banerjee and Ted Pedersen. An adapted lesk algorithm for word sense disambiguation using WordNet. In *Computational Linguistics and Intelligent Text Processing, Third International Conference (CICLing)*, pages 136–145, 2002.
- [BPS⁺05] Stephan Bloehdorn, Kosmas Petridis, Carsten Saathoff, Nikos Simou, Vassilis Tzouvaras, Yannis S. Avrithis, Siegfried Handschuh, Ioannis Kompatsiaris, Steffen Staab, and Michael G. Strintzis. Semantic annotation of images and videos for multimedia analysis. In *The Semantic Web: Research and Applications, Second European Semantic Web Conference (ESWC)*, pages 592–607, 2005.
- [BS84] Bruce G. Buchanan and Edward H. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, MA, 1984.
- [BS99] Brigitte Biebow and Sylvie Szulman. Terminae: a method and a tool to build a domain ontology. In *Proceedings of the 11th European Workshop on Knowledge Acquisition, Modelling and Management*, pages 49–66, 1999.
- [BTMS04] Michael Bada, Daniele Turi, Robin McEntire, and Robert Stevens. Using reasoning to guide annotation with gene ontology terms in goat. *SIGMOD Record*, 33(2):27–32, 2004.

REFERENCES

- [BYRN99] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.
- [Cas90] George Casella. *Statistical Inference*. Brooks/Cole Publish Co., 1990.
- [CBB⁺04] Jesús Contreras, V. Richard Benjamins, Mercedes Blázquez, Silvestre Losada, R. Salla, J. Sevilla, D. Navarro, Joaquín Casillas, A. Mompó, Diego Patón, Óscar Corcho, P. Tena, and I. Martos. A semantic portal for the international affairs sector. In *Engineering Knowledge in the Age of the Semantic Web, 14th International Conference (EKAW)*, pages 203–215, 2004.
- [CDI98] Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 307–318, 1998.
- [CFM06] Calero Coral, Ruiz Francisco, and Piattini Mario. *Ontologies for Software Engineering and Software Technology*. Springer-Verlag, Berlin, Heidelberg, 2006.
- [CFV07] Pablo Castells, Miriam Fernández, and David Vallet. An adaptation of the vector-space model for ontology-based information retrieval. *IEEE Trans. Knowl. Data Eng.*, 19(2):261–272, 2007.
- [CGG92] Jim Cowie, Joe Guthrie, and Louise Guthrie. Lexical disambiguation using simulated annealing. In *Proceedings of the 14th conference on Computational linguistics (COLING)*, pages 359–365, 1992.
- [CGR⁺05] Massimiliano Ciaramita, Aldo Gangemi, Esther Ratsch, Jasmin Saric, and Isabel Rojas. Unsupervised learning of semantic relations between concepts of a molecular biology ontology. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 659–664, 2005.
- [CNM04] Aaron Clauset, Mark E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70:066111, 2004.

REFERENCES

- [CNPk05] Paul-Alexandru Chirita, Wolfgang Nejdl, Raluca Paiu, and Christian Kohlschütter. Using odp metadata to personalize search. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 178–185, 2005.
- [CV05] Philipp Cimiano and Johanna Völker. Text2onto. In *Natural Language Processing and Information Systems, 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, pages 227–238, 2005.
- [Dai96] Béatrice Daille. Study and implementation of combined techniques for automatic extraction of terminology. In Judith Klavans and Philip Resnik, editors, *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, pages 49–66. The MIT Press, 1996.
- [DDS03] John Davies, Alistair Duke, and York Sure. Ontoshare: a knowledge management environment for virtual communities of practice. In *Proceedings of the 2nd International Conference on Knowledge Capture (K-CAP)*, pages 20–27, 2003.
- [DFJ⁺04] Li Ding, Timothy W. 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 2004 ACM CIKM International Conference on Information and Knowledge Management (CIKM)*, pages 652–659, 2004.
- [DJZ⁺09] Ying Ding, Elin K. Jacob, Zhixiong Zhang, Schubert Foo, Erjia Yan, Nicolas L. George, and Lijiang Guo. Perspectives on social tagging. *Journal of the American Society for Information Science and Technology (JASIST)*, 60(12):2388–2401, 2009.
- [DKL07] Hoa Trang Dang, Diane Kelly, and Jimmy Lin. Overview of the trec 2007 question answering track. In *Proceedings of The Sixteenth Text REtrieval Conference, (TREC)*, 2007.
- [Dun93] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

REFERENCES

- [DW04] John Davies and Richard Weeks. QuizRDF: Search technology for the semantic web. In *37th Hawaii International Conference on System Sciences (HICSS)*, pages 40112–40119, 2004.
- [ECD⁺04] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th international conference on World Wide Web (WWW)*, pages 100–110, 2004.
- [Eng03] Robert Engels. Ontology extraction tool, deliverable 6. Technical report, Ontoknowledge, 2003. <http://www.ontoknowledge.org/del.shtml>.
- [Euz04] Jérôme Euzenat. An api for ontology alignment. In *International Semantic Web Conference (ISWC)*, pages 698–712, 2004.
- [Fan61] Robert M. Fano. *Transmission of Information*. MIT Press: Cambridge MA, 1961.
- [FCJ01] Yongjian Fu, Mario Creado, and Chunhua Ju. Reorganizing web sites based on user access patterns. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management (CIKM)*, pages 583–585, 2001.
- [Fel98] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT press, 1998.
- [FFR97] Adam Farquhar, Richard Fikes, and James Rice. The ontolingua server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46(6):707–727, 1997.
- [FGL08] Jianping Fan, Yuli Gao, and Hangzai Luo. Integrating concept ontology and multitask learning to achieve more effective classifier training for multilevel image annotation. *IEEE Transactions on Image Processing*, 17(3):407–426, 2008.
- [FN98] David Faure and Claire Nédellec. A corpus-based conceptual clustering method for verb frames and ontology acquisition. In *LREC workshop on Adapting lexical and corpus resources to sublanguages and applications*, 1998.

REFERENCES

- [GC02] Shayan Ghazizadeh and Sudarshan S. Chawathe. Seus: Structure extraction using summaries. In *Discovery Science, 5th International Conference (DS)*, pages 71–85, 2002.
- [GF95] Michael Grüninger and Mark S. Fox. Methodology for the design and evaluation of ontologies. In *IJCAI’95, Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.
- [GGA⁺02] Pablo Gamallo, Marco Gonzalez, Alexandre Agustini, Gabriel Lopes, and Vera S. De Lima. Mapping syntactic dependencies onto semantic relations. In *ECAI Workshop on Machine Learning and Natural Language Processing for Ontology Engineering*, pages 15–22, 2002.
- [GJ79] M. R. Garey and David S. Johnson. *Computer and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [GL90] Ramanathan V. Guha and Douglas B. Lenat. Cyc: A midterm report. *AI Magazine*, 11(3):32–59, 1990.
- [GMM03] Ramanathan V. Guha, Rob McCool, and Eric Miller. Semantic search. In *Proceedings of the Twelfth International World Wide Web Conference (WWW)*, pages 700–709, 2003.
- [GMV99] Nicola Guarino, Claudio Masolo, and Guido Vetere. Ontoseek: Content-based access to the web. *IEEE Intelligent Systems*, 14(3):70–80, 1999.
- [GN87] Michael R. Genesereth and Nils J. Nilsson. *Logical foundations of artificial intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.
- [GNV03] Aldo Gangemi, Roberto Navigli, and Paola Velardi. The OntoWordNet Project: extension and axiomatization of conceptual relations in WordNet. *Lecture Notes in Computer Science*, pages 820–838, 2003.
- [GPCFL04] Asuncion Gomez-Perez, Oscar Corcho, and Mariano Fernandez-Lopez. *Ontological Engineering : with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. First Edition (Advanced Information and Knowledge Processing)*. July 2004.

REFERENCES

- [GPMM] A. Gomez-Perez and D. Manzano-Macho. Technical report, OntoWeb. <http://www.ontoweb.org/Members/ruben/Deliverable1.5>.
- [Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.
- [GSBS03] Lin Guo, Feng Shao, Chavdar Botev, and Jayavel Shanmugasundaram. Xrank: Ranked keyword search over xml documents. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 16–27, 2003.
- [GSY07] Lin Guo, Jayavel Shanmugasundaram, and Golan Yona. Topology search over biological databases. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE)*, pages 556–565, 2007.
- [Gua98] Nicola Guarino. Formal ontology and information systems. In *Proceedings of the First International Conference on Formal Ontologies in Information Systems (FOIS)*, pages 3–15, 1998.
- [GW97] Roy Goldman and Jennifer Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *Proceedings of 23rd International Conference on Very Large Data Bases (VLDB)*, pages 436–445, 1997.
- [HCD94] Lawrence B. Holder, Diane J. Cook, and Surnjani Djoko. Substructure discovery in the subdue system. In *Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop*, pages 169–180, 1994.
- [hCPCP01] Ed Huai hsin Chi, Peter Pirolli, Kim Chen, and James E. Pitkow. Using information scent to model user information needs and actions and the web. In *Proceedings of the SIG-CHI on Human factors in computing systems (CHI)*, pages 490–497, 2001.
- [Hea92] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 539–545, 1992.

REFERENCES

- [Hea98] Marti A. Hearst. Automated discovery of WordNet relations. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*, chapter 5, pages 132–152. MIT press, 1998.
- [HH00] Jeff Heflin and James Hendler. Searching the web with shoe. In *AAAI-2000 Workshop on AI for Web Search*, pages 35–40, 2000.
- [HHF03] Yulan He, Siu Cheung Hui, and Alvis Cheuk M. Fong. Citation-based retrieval for scholarly publications. *IEEE Intelligent Systems*, 18(2):58–65, 2003.
- [HHSTS05] Peter Haase, Andreas Hotho, Lars Schmidt-Thieme, and York Sure. Collaborative and usage-driven evolution of personal ontologies. In *The Semantic Web: Research and Applications, Second European Semantic Web Conference (ESWC)*, pages 486–499, 2005.
- [HMS⁺05] Eero Hyvönen, Eetu Mäkelä, Mirva Salminen, Arttu Valo, Kim Viljanen, Samppa Saarela, Miikka Junnila, and Suvi Kettula. Museumfinland – finnish museums on the semantic web. *Journal of Web Semantics*, 3(2-3):224–241, 2005.
- [HWAMAS04] Christian Halaschek-Wiener, Boanerges Aleman-Meza, Ismailcem Budak Arpinar, and Amit P. Sheth. Discovering and ranking semantic associations over a large rdf metabase. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB)*, pages 1317–1320, 2004.
- [HWGS⁺06] Christian Halaschek-Wiener, Jennifer Golbeck, Andrew Schain, Michael Grove, Bijan Parsia, and James A. Hendler. Annotation and provenance tracking in semantic web photo libraries. In *Provenance and Annotation of Data, International Provenance and Annotation Workshop (IPAW)*, pages 82–89, 2006.
- [Ive84] Gudmund R. Iversen. *Bayesian statistical inference*. Sage Publications, Inc, 1984.

REFERENCES

- [IWM00] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Principles of Data Mining and Knowledge Discovery, 4th European Conference (PKDD)*, pages 13–23, 2000.
- [JLM03] Jiwoon Jeon, Victor Lavrenko, and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 119–126, 2003.
- [Joh94] Paul Johannesson. A method for transforming relational schemas into conceptual schemas. In *Proceedings of the Tenth International Conference on Data Engineering (ICDE)*, pages 190–201, 1994.
- [JT05] Xing Jiang and Ah-Hwee Tan. Mining ontological knowledge from domain-specific text documents. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, pages 665–668, 2005.
- [JT06a] Xing Jiang and Ah-Hwee Tan. Learning and inferencing in user ontology for personalized semantic web services. In *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, pages 1067–1068, 2006.
- [JT06b] Xing Jiang and Ah-Hwee Tan. Ontosearch: A full-text search engine for the semantic web. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 1325–1330, 2006.
- [JT09] Xing Jiang and Ah-Hwee Tan. Learning and inferencing in user ontology for personalized semantic web search. *Information Sciences*, 179(16):2794–2808, 2009.
- [JT10] Xing Jiang and Ah-Hwee Tan. Crcol: A semantic-based domain ontology learning system. *Journal of the American Society for Information Science and Technology (JASIST)*, 61(1):150–168, 2010.

REFERENCES

- [JXWT09] Xing Jiang, Hui Xiong, Chen Wang, and Ah-Hwee Tan. Mining globally distributed frequent subgraphs in a single labeled graph. *Data & Knowledge Engineering (DKE)*, 68(10):1034–1058, 2009.
- [Kas99] Vipul Kashyap. Design and creation of ontologies for environmental information retrieval. In *Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management*, 1999.
- [KF01] Michel C. A. Klein and Dieter Fensel. Ontology versioning on the semantic web. In *Proceedings of SWWS’01, The first Semantic Web Working Symposium*, pages 75–91, 2001.
- [KFKO02] Michel C. A. Klein, Dieter Fensel, Atanas Kiryakov, and Damyan Ognyanov. Ontology versioning and change detection on the web. In *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference (EKAW)*, pages 197–212, 2002.
- [KK01] Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM)*, pages 313–320, 2001.
- [KK05] Michihiro Kuramochi and George Karypis. Finding frequent patterns in a large sparse graph. *Data Min. Knowl. Discov.*, 11(3):243–271, 2005.
- [KKS⁺06] Mehmet Koyutürk, Yohan Kim, Shankar Subramaniam, Wojciech Szpankowski, and Ananth Grama. Detecting conserved interaction patterns in biological networks. *Journal of Computational Biology*, 13(7):1299–1322, 2006.
- [KM02] Dan Klein and Christopher D. Manning. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15, NIPS 2002*, pages 3–10, 2002.
- [KMH04] Latifur Khan, Dennis McLeod, and Eduard H. Hovy. Retrieval effectiveness of an ontology-based model for information selection. *VLDB J.*, 13(1):71–85, 2004.

REFERENCES

- [KÖ08] Mustafa Kirac and Gultekin Özsoyoglu. Protein function prediction based on patterns in biological networks. In *Research in Computational Molecular Biology, 12th Annual International Conference (RECOMB)*, pages 197–213, 2008.
- [KPO⁺03] Atanas Kiryakov, Borislav Popov, Damyan Ognyanoff, Dimitar Manov, Angel Kirilov, and Miroslav Goranov. Semantic annotation, indexing, and retrieval. In *International Semantic Web Conference (ISWC)*, pages 484–499, 2003.
- [LD94] Nada Lavrac and Saso Dzeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, 1994.
- [Les86] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, pages 24–26, 1986.
- [LG89] Douglas B. Lenat and Ramanathan V. Guha. *Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
- [LGPSS99] Mariano Fernández López, Asunción Gómez-Pérez, Juan Pazos Sierra, and Alejandro Pazos Sierra. Building a chemical ontology using methontology and the ontology design environment. *IEEE Intelligent Systems*, 14(1):37–46, 1999.
- [LMMP96] Wai Lam, Snehasis Mukhopadhyay, Javed Mostafa, and Mathew J. Palakal. Detection of shifts in user interests for personalized information filtering. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 317–325, 1996.
- [LS04] Hugo Liu and Push Singh. Conceptnet: a practical commonsense reasoning toolkit. *BT Technology Journal*, 22(4):211–226, 2004.
- [LZ06] Yuefeng Li and Ning Zhong. Mining ontology for automatically acquiring web user information needs. *IEEE Trans. Knowl. Data Eng.*, 18(4):554–568, 2006.

REFERENCES

- [MISA04] Peter Mika, Victor Iosif, York Sure, and Hans Akkermans. Ontology-based content management in a virtual organization. In *Handbook on Ontologies*, pages 455–476. 2004.
- [MMS⁺03] Alexander Maedche, Boris Motik, Ljiljana Stojanovic, Rudi Studer, and Raphael Volz. Ontologies for enterprise knowledge management. *IEEE Intelligent Systems*, 18(2):26–33, 2003.
- [MN04] Dan I. Moldovan and Adrian Novischi. Word sense disambiguation of wordnet glosses. *Computer Speech & Language*, 18(3):301–317, 2004.
- [MNV02] Michele Missikoff, Roberto Navigli, and Paola Velardi. The usable ontology: An environment for building and assessing a domain ontology. In *International Semantic Web Conference (ISWC)*, pages 39–53, 2002.
- [MS00] Alexander Maedche and Steffen Staab. Mining ontologies from text. In *Knowledge Acquisition, Modeling and Management, 12th International Conference (EKAW)*, pages 189–202, 2000.
- [MS01] Alexander Maedche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.
- [MSR04] Stuart E. Middleton, Nigel R. Shadbolt, and David C. De Roure. Ontological user profiling in recommender systems. *ACM Trans. Inf. Syst.*, 22(1):54–88, 2004.
- [MVF03] Michele Missikoff, Paola Velardi, and Paolo Fabriani. Text mining techniques to automatically enrich a domain ontology. *Appl. Intell.*, 18(3):323–340, 2003.
- [MZ02] Alexander Maedche and Valentin Zacharias. Clustering ontology-based metadata in the semantic web. In *Principles of Data Mining and Knowledge Discovery, 6th European Conference (PKDD)*, pages 348–360, 2002.
- [Nak02] Chieko Nakabasami. An inductive approach to assertional mining of web ontology revision. In *RuleML 2002, Proceedings of the International Workshop on Rule Markup Languages for Business Rules on the Semantic Web*, 2002.

REFERENCES

- [NFM00] Natalya Fridman Noy, Ray W. Ferguson, and Mark A. Musen. The knowledge model of protégé-2000: Combining interoperability and flexibility. In *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW)*, pages 17–32, 2000.
- [NKKM04] Natalya Fridman Noy, Sandhya Kunnatur, Michel C. A. Klein, and Mark A. Musen. Tracking changes during ontology evolution. In *International Semantic Web Conference (ISWC)*, pages 259–273, 2004.
- [NLHD08] Vít Nováček, Loredana Laera, Siegfried Handschuh, and Brian Davis. Infrastructure for dynamic knowledge integration - automated biomedical ontology extension using textual resources. *Journal of Biomedical Informatics*, 41(5):816–828, 2008.
- [NP03] Ian Niles and Adam Pease. Linking lexicons and ontologies: Mapping wordnet to the suggested upper merged ontology. In *Proceedings of the International Conference on Information and Knowledge Engineering (IKE)*, pages 412–416, 2003.
- [NT04] Ian Niles and Allan Terry. The milo: A general-purpose, mid-level ontology. In *Proceedings of the International Conference on Information and Knowledge Engineering (IKE)*, pages 15–19, 2004.
- [Ö02] Patric R. J. Östergård. A fast algorithm for the maximum clique problem. *Discrete Appl. Math.*, 120(1-3):197–207, 2002.
- [OBHG03] Daniel Oberle, Bettina Berendt, Andreas Hotho, and Jorge Gonzalez. Conceptual user tracking. In *Web Intelligence, First International Atlantic Web Intelligence Conference*, pages 155–164, 2003.
- [OWL] Web ontology language (owl). <http://www.w3.org/2004/OWL/>.
- [PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.
- [PBTk06] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the*

REFERENCES

- 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (ACL)*, pages 433–440, 2006.
- [PG99] Alexander Pretschner and Susan Gauch. Ontology based personalized search. In *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 391–398, 1999.
- [PNL02] Adam Pease, Ian Niles, and John Li. The suggested upper merged ontology: A large ontology for the semantic web and its applications. In *Workshop on Ontologies and the Semantic Web, held with AAAI2002*, 2002.
- [PS06] Eric Prud’hommeaux and Andy Seaborne. SPARQL Query Language for RDF. Technical report, 2006.
- [RDB⁺06] Daniel L. Rubin, Olivier Dameron, Yasser Bashir, David Grossman, Parvati Dev, and Mark A. Musen. Using ontologies linked with geometric models to reason about penetrating injuries. *Artificial Intelligence in Medicine*, 37(3):167–176, 2006.
- [RDF] RDF Schema Specification. <http://www.w3c.org/TR/rdf-schema/>.
- [RHO⁺02] Daniel L. Rubin, Micheal Hewett, Diane E. Oliver, Teri E. Klein, and Russ B. Altman. Automating data acquisition into ontologies from pharmacogenetics relational data sources using declarative object definitions and xml. In *Pacific Symposium on Biocomputing (PSB)*, pages 88–99, 2002.
- [Rie00] Doug Riecken. introduction: Personalized views of personalization. *Commun. ACM*, 43(8):26–28, August 2000.
- [Rig98] German Rigau. *Automatic Acquisition of Lexical Knowledge from MRDs*. PhD thesis, Dept. LSI, Universitat Politcnica de Catalunya, 1998.
- [Rij79] C. J. Van Rijsbergen. *Information Retrieval*. Butterworths, London, 2nd edition, 1979.

REFERENCES

- [RM00] Bradley J. Rhodes and Pattie Maes. Just-in-time information retrieval agents. *IBM Systems Journal*, 39(3&4):685–704, 2000.
- [RSdA04] Cristiano Rocha, Daniel Schwabe, and Marcus Poggi de Aragão. A hybrid approach for searching in the semantic web. In *Proceedings of the 13th international conference on World Wide Web (WWW)*, pages 374–383, 2004.
- [RT03] Kanagasabai Rajaraman and Ah-Hwee Tan. Mining semantic networks for knowledge discovery. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM)*, pages 633–636, 2003.
- [RUSB07] J. Ramanand, Akshay Ukey, Brahm Kiran Singh, and Pushpak Bhattacharyya. Mapping and structural analysis of multi-lingual wordnets. *IEEE Data Eng. Bull.*, 30(1):30–44, 2007.
- [SA95] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In *Proceedings of 21th International Conference on Very Large Data Bases (VLDB)*, pages 407–419, 1995.
- [SAKY98] Rieko Sekiuchi, Chizuru Aoki, Masaki Kurematsu, and Takahira Yamaguchi. Doodle: A domain ontology rapid development environment. In *5th Pacific Rim International Conference on Artificial Intelligence, (PRICAI)*, pages 194–204, 1998.
- [SDM80] Juan C Sagar, David Dungworth, and Peter F McDonald. *English Special Language: Principles and Practice in science and technology*. Wiesbaden: Oscar Brandstetter Verlag KG, 1980.
- [SEA⁺02] York Sure, Michael Erdmann, Jürgen Angele, Steffen Staab, Rudi Studer, and Dirk Wenke. Ontoedit: Collaborative ontology development for the semantic web. In *International Semantic Web Conference (ISWC)*, pages 221–235, 2002.
- [Sea04] Andy Seaborne. RDQL - A Query Language for RDF. Technical report, 2004.

REFERENCES

- [SFJ02] Urvi Shah, Timothy W. Finin, and Anupam Joshi. Information retrieval on the semantic web. In *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management (CIKM)*, pages 461–468, 2002.
- [SK02] Steffen Schulze-Kremer. Ontologies for molecular biology and bioinformatics. In *Silico Biology*, 2:17, 2002.
- [SKW07] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web (WWW)*, pages 697–706, 2007.
- [SM86] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.
- [Sow99] John F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Course Technology, August 1999.
- [SPL07] Peter Scheir, Viktoria Pammer, and Stefanie N. Lindstaedt. Information retrieval on the semantic web - does it exist? In *LWA 2007: Lernen - Wissen - Adaption, Halle, September 2007, Workshop Proceedings*, pages 252–257, 2007.
- [SRKR97] B. Swartout, P. Ramesh, K. Knight, and T. Russ. Toward distributed use of large-scale ontologies. In *Proceeding of AAAI Spring Symposium Series, Workshop on Ontological Engineering*, pages 33–40, 1997.
- [SSGS03] Ljiljana Stojanovic, Nenad Stojanovic, Jorge Gonzalez, and Rudi Studer. Ontomanager - a system for the usage-based ontology management. In *CoopIS/DOA/ODBASE*, pages 858–875, 2003.
- [SSM03] Malcolm Slaney, Jayashree Subrahmonia, and Paul P. Maglio. Modeling multitasking users. In *User Modeling 2003, 9th International Conference*, pages 188–197, 2003.
- [SSS03] Nenad Stojanovic, Rudi Studer, and Ljiljana Stojanovic. An approach for the ranking of query results in the semantic web. In *International Semantic Web Conference (ISWC)*, pages 500–516, 2003.

REFERENCES

- [SSSS01] Steffen Staab, Rudi Studer, Hans-Peter Schnurr, and York Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1):26–34, 2001.
- [SSV02] Ljiljana Stojanovic, Nenad Stojanovic, and Raphael Volz. Migrating data-intensive web sites into the semantic web. In *Proceedings of the 2002 ACM symposium on Applied computing (SAC)*, pages 1100–1107, 2002.
- [Sto04] Ljiljana Stojanovic. *Methods and Tools for Ontology Evolution*. PhD thesis, Universität Karlsruhe (TH), Universität Karlsruhe (TH), Institut AIFB, D-76128 Karlsruhe, 2004.
- [SW97] Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *J. ACM*, 44(4):585–591, 1997.
- [TB99] Lance Tokuda and Don S. Batory. Automating three modes of evolution for object-oriented software architectures. In *Proceedings of the 5th USENIX Conference on Object-Oriented Technologies & Systems (COOTS)*, pages 189–202, 1999.
- [TBRH08] Volker Tresp, Markus Bundschuh, Achim Rettinger, and Yi Huang. Towards machine learning on the semantic web. *Uncertainty Reasoning for the Semantic Web I: ISWC International Workshops, URSW 2005-2007, Revised Selected and Invited Papers*, pages 282–314, 2008.
- [TKMS03] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL)*, pages 173–180, 2003.
- [TLZ09] Xiaohui Tao, Yuefeng Li, and Ning Zhong. A personalized ontology model for web information gathering. *IEEE Trans. Knowl. Data Eng. (accepted)*, 2009.
- [TLZN07] Xiaohui Tao, Yuefeng Li, Ning Zhong, and Richi Nayak. Ontology mining for personalized web information gathering. In *2007 IEEE / WIC / ACM International Conference on Web Intelligence (WI)*, pages 351–358, 2007.

REFERENCES

- [TT98] Ah-Hwee Tan and Christine Teo. Learning user profiles for personalized information dissemination. In *International Joint Conference on Neural Networks (IJCNN)*, pages 183–188, 1998.
- [UG04] Michael Uschold and Michael Grüninger. Ontologies and semantics for seamless connectivity. *SIGMOD Record*, 33(4):58–64, 2004.
- [UK95] Mike Uschold and Martin King. Towards a methodology for building ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95*, 1995.
- [VCF⁺07] David Vallet, Pablo Castells, Miriam Fernández, Phivos Mylonas, and Yannis S. Avrithis. Personalized content retrieval in context using ontological knowledge. *IEEE Trans. Circuits Syst. Video Techn.*, 17(3):336–346, 2007.
- [VGS02] Natalia Vanetik, Ehud Gudes, and Solomon Eyal Shimony. Computing frequent graph patterns from semistructured data. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM)*, pages 458–465, 2002.
- [Voo93] Ellen M. Voorhees. Using WordNet to disambiguate word senses for text retrieval. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 171–180, 1993.
- [Wat03] Duncan J. Watts. *Small Worlds : The Dynamics of Networks between Order and Randomness (Princeton Studies in Complexity)*. Princeton University Press, 2003.
- [WHLS06] Junmei Wang, Wynne Hsu, Mong-Li Lee, and Chang Sheng. A partition-based approach to graph mining. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE)*, page 74, 2006.
- [WJCT08] Huan Wang, Xing Jiang, Liang-Tien Chia, and Ah-Hwee Tan. Ontology enhanced web image retrieval: aided by wikipedia & spreading activation theory. In *Proceedings of the 1st ACM SIGMM International Conference on Multimedia Information Retrieval, MIR 2008, Vancouver, British Columbia, Canada, October 30-31, 2008*, pages 195–201, 2008.

REFERENCES

- [WS03] Song Wang and Jeffrey Mark Siskind. Image segmentation with ratio cut. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(6):675–690, 2003.
- [WW08] Fei Wu and Daniel S. Weld. Automatically refining the wikipedia infobox ontology. In *Proceedings of the 17th International Conference on World Wide Web (WWW)*, pages 635–644, 2008.
- [WWP⁺04] Chen Wang, Wei Wang, Jian Pei, Yongtai Zhu, and Baile Shi. Scalable mining of large disk-based graph databases. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 316–325, 2004.
- [XKPS02] Feiyu Xu, Daniela Kurz, Jakub Piskorski, and Sven Schmeier. A domain adaptive approach to automatic acquisition of domain relevant terms and their relations with bootstrapping. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC)*, 2002.
- [XZZ⁺08] Changsheng Xu, Yi-Fan Zhang, Guangyu Zhu, Yong Rui, Hanqing Lu, and Qingming Huang. Using webcast text for semantic event detection in broadcast sports video. *Multimedia, IEEE Transactions on*, 10(7):1342–1355, Nov. 2008.
- [YH02] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM)*, pages 721–724, 2002.
- [YM95] Kenichi Yoshida and Hiroshi Motoda. Clip: Concept learning from inference patterns. *Artif. Intell.*, 75(1):63–92, 1995.
- [ZTC07] Dimitris Zeginis, Yannis Tzitzikas, and Vassilis Christophides. On the foundations of computing deltas between rdf models. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference (ISWC/ASWC)*, pages 637–651, 2007.
- [ZYZ⁺05] Lei Zhang, Yong Yu, Jian Zhou, Chenxi Lin, and Yin Yang. An enhanced model for searching in semantic portals. In *Proceedings of the 14th international conference on World Wide Web (WWW)*, pages 453–462, 2005.