

**KNOWLEDGE MANAGEMENT USING MACHINE
LEARNING, NATURAL LANGUAGE PROCESSING
AND ONTOLOGY**

A thesis

submitted to the

University of Wales

for the degree of

Doctor of Philosophy

by

Qiao Tang

Cardiff School of Engineering

University of Wales, Cardiff

February 2006

UMI Number: U584821

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI U584821

Published by ProQuest LLC 2013. Copyright in the Dissertation held by the Author.
Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against
unauthorized copying under Title 17, United States Code.



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

SUMMARY

This research developed a concept indexing framework which systematically integrates machine learning, natural language processing and ontology technologies to facilitate knowledge acquisition, extraction and organisation.

The research reported in this thesis focuses first on the conceptual model of concept indexing, which represents knowledge as entities and concepts. Then the thesis outlines its benefits and the system architecture using this conceptual model.

Next, the thesis presents a knowledge acquisition framework using machine learning in focused crawling Web content to enable automatic knowledge acquisition.

Then, the thesis presents two language resources developed to enable ontology tagging, which are: an ontology dictionary and an ontologically tagged corpus. The ontologically tagged corpus is created using a heuristic algorithm developed in the thesis.

Next, the ontology tagging algorithm is developed with the ontology dictionary and the ontologically tagged corpus to enable ontology tagging.

Finally, the thesis presents the conceptual model, the system architecture, and the prototype system using concept indexing developed to facilitate knowledge acquisition, extraction and organisation.

The solutions proposed in the thesis are illustrated with examples based on a prototype system developed in this thesis. This work was carried out in the Cardiff University partially funded through the Seedcorn Grant Scheme.

ACKNOWLEDGEMENTS

I would like to thank the supervisors of my studies, Dr Rossi Setchi and Professor Stefan Dimov, for their invaluable guidance and support throughout my work.

Without their support I could not reach this far.

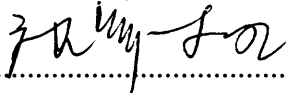
All members of the I2S group and Intelligent Systems Laboratory are thanked for their friendships and help.

My deepest gratitude is to my family who has given continuous support and encouragement to me.

DECLARATION AND STATEMENTS

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

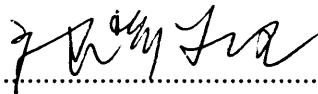
Signed..........(Q. Tang - Candidate)

Date.....12/05/2006.....

Statement 1

This thesis is the result of my own investigations, except where otherwise stated.

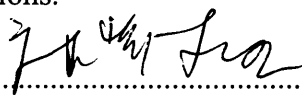
Other sources are acknowledged by giving explicit references. A list of references is appended.

Signed..........(Q. Tang - Candidate)


Date.....12/05/2006.....

Statement 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed..........(Q. Tang - Candidate)

Date.....12/05/2006.....

Signed..........(Dr Rossi Setchi - Supervisor)

Date.....12/05/06.....

TABLE OF CONTENTS

SUMMARY	i
ACKNOWLEDGEMENTS	ii
DECLARATION AND STATEMENTS	iii
LIST OF FIGURES	x
LIST OF TABLES	xiii
ABBREVIATIONS	xv
CHAPTER 1. INTRODUCTION	1
1.1 Motivation	1
1.2 Aims and Objectives	4
1.3 Outline	5
CHAPTER 2. REVIEW OF TECHNOLOGY APPROACHES TO KNOWLEDGE MANAGEMENT	9
2.1 KNOWLEDGE MANAGEMENT	9
2.1.1 Knowledge	9
2.1.2 Knowledge Management	10
2.2 INFORMATION RETRIEVAL	11
2.2.1 Information Retrieval and Web Search	11
2.2.2 Crawling	13
2.2.3 Models for Information Retrieval	16
2.2.4 Indexing	19
2.2.5 Storage and Query	20
2.2.6 Applications of Information Retrieval	20
2.3 MACHINE LEARNING TECHNIQUES	21
2.3.1 Support Vector Machine	21
2.3.2 Rule-Based Learning	22
2.3.3 Memory-Based Learning	23

2.4 NATURAL LANGUAGE PROCESSING	24
2.4.1 Part of Speech Tagging	24
2.4.2 Word Sense Disambiguation	25
2.4.3 Concordance and Collocations	25
2.4.4 Concept Indexing	26
2.4.5 Information Extraction	27
2.5 THE SEMANTIC WEB AND ONTOLOGY	28
2.5.1 Ontology Engineering	28
2.5.2 Information Extraction for the Semantic Web	29
2.6 DISCUSSION	30
2.7 SUMMARY	32
CHAPTER 3. FORMAL REPRESENTATION OF CONTENT USING	33
CONCEPT INDEXING	
3.1 CONCEPT INDEXING	33
3.1.1 Conventional Models for Knowledge Representation	33
3.1.2 Requirements for Knowledge Representation of Explicit Knowledge	35
3.1.3 Definitions	39
3.1.4 Mathematical Model	42
3.1.5 Benefits of Adopting Concept Indexing	45
3.2 A FRAMEWORK FOR CONCEPT INDEXING	48
3.3 ILLUSTRATIVE EXAMPLE	51
3.4 A SYSTEM FOR CONCEPT INDEXING	54
3.4.1 System Architecture	54
3.4.2 Main Processes	55
3.5 SUMMARY	57
CHAPTER 4. AUTOMATIC INFORMATION ACQUISITION USING	59
MACHINE LEARNING	
4.1 FINDING INFORMATION ON THE WEB	59
4.2 AN INFORMATION ACQUISITION FRAMEWORK	60

4.2.1 Structure of the Information Acquisition Framework	60
4.2.2 Information Acquisition Process	62
4.3 AN IMPROVED INFORMATION ACQUISITION FRAMEWORK	65
4.3.1 Using Support Vector Machine (SVM) for Information Retrieval	65
4.3.2 Machine Learning in the Improved Information Acquisition Framework	67
4.3.3 Feature Selection for the Proposed Machine Learning Algorithm	70
4.3.4 Case Studies	73
General Approach	73
Case Study 1: Fishing	76
Case Study 2: Site Engineering	81
Discussion	84
4.4 SYSTEM DESIGN AND FUNCTION DESIGN	84
4.5 SUMMARY	88
CHAPTER 5. ONTOLOGY DICTIONARY AND CORPUS FOR ONTOLOGY TAGGING	89
5.1 ONTOLOGY TAGGING	89
5.2 BUILDING AN ONTOLOGY DICTIONARY	92
5.2.1 Selecting Dictionaries	92
5.2.2 Dictionaries	93
WordNet	94
Roget's Thesaurus	95
5.2.3 Comparisons between Roget's Thesaurus and WordNet	97
5.2.4 Building an Ontology Dictionary from Roget's Thesaurus	99
5.3 BUILDING AN ONTOLOGICALLY TAGGED CORPUS	103
5.3.1 Method of Building an Ontologically Tagged Corpus	103
5.3.2 Building a Machine Readable Dictionary from WordNet	104
5.3.3 A Heuristic Approach for Semantic Mapping Between eWord and OntoRo	107

General Approach	107
Testing of the Semantic Mapping Algorithm	118
Discussion	120
5.3.4 Converting the Semcor Corpus into an Ontologically Tagged Corpus	121
5.4 SUMMARY	124
CHAPTER 6. FULL TEXT ONTOLOGY TAGGING BASED ON	125
MACHINE LEARNING	
6.1 ONTOLOGY TAGGING ALGORITHM	125
6.1.1 Statistical and Context Information	125
6.1.2 Variables	129
6.1.3 Statistical Rules and Context Rules	132
Statistical Rules	132
Context Rules	133
6.1.4 Training and Tagging with the Ontology Tagging Algorithm	134
Training	134
Tagging	136
6.2 CASE STUDIES	137
6.2.1 General Approach	137
6.2.2 Case Study 1	140
6.2.3 Case Study 2	141
6.2.4 Case Study 3	143
6.2.5 Discussion	147
6.3 SUMMARY	148
CHAPTER 7 KNOWLEDGE MANAGEMENT SYSTEM BASED ON	149
ENTITY AND CONCEPT INDEXING	
7.1 ENTITY AND CONCEPT INDEXING	149
7.1.1 Conceptual Model	149
7.1.2 Entity Tagging	152
7.1.3 Benefits	153

7.2 PROCESSING	154
7.3 IMPLEMENTATION AND TESTING	157
7.3.1 Indexing Tests	157
7.3.2 Testing of Entity and Concept Extraction	158
7.4 SEARCH SPEED COMPARISON WITH MYSQL	165
7.5 CASE STUDIES	168
7.5 DISCUSSIONS	188
7.6 SUMMARY	195
CHAPTER 8. CONTRIBUTIONS, CONCLUSIONS AND FUTURE	196
WORK	
8.1 CONTRIBUTIONS	196
8.2 CONCLUSIONS	198
8.3 IMPLEMENTATION WORK	200
8.4 FUTURE WORK	204
APPENDIX A. MATERIALS AND RESULTS	206
A.1 POSITIVE AND NEGATIVE EXAMPLES	206
A.2 TRAINING EXAMPLE 1: AN HTML WEB PAGE (FRAGMENT)	207
A.3 TRAINING EXAMPLE 1: AN HTML WEB PAGE STRIPPED TO TEXT ONLY (FRAGMENT)	208
A.4 SMALL DICTIONARY	209
A.5 DATA FILE GENERATED BY SVM FROM THE TRAINING SET (FRAGMENT)	211
A.6 TESTING RESULTS FOR CASE STUDY 1	212
A.7 TESTING RESULTS FOR CASE STUDY 2	214
APPENDIX B. SAMPLE SOURCE CODE FOR CHAPTER 4	216
APPENDIX C. SEMCOR, WORDNET AND ROGET'S	229
C.1 TAGS USED IN SEMCOR 1.6	229
C.2 SEMCOR ANNOTATED TEXT SEMPLE	230
C.3 WORDNET AND ROGET'S ENTRY SAMPLES	231

APPENDIX D. SAMPLE SOURCE CODE FOR CHAPTER 5	232
D.1 SAMPLE CODE FOR SEMANTIC MAPPPING	232
D.2 SAMPLE CODE FOR SEMCOR ANNOTATED CORPUS PARSER	234
APPENDIX E. SAMPLE SOURCE CODE FOR CHAPTER 6	238
APPENDIX F. SAMPLE SOURCE CODE FOR CHAPTER 7	241
APPENDIX G. DATA OF TESTING RESULTS USED IN CHAPTER 7	248
G.1 INDEXING TESTS	248
G.2 TESTING OF ENTITY AND CONCEPT EXTRACTION	250
G.3 RETRIEVAL SPEED TESTS	252
REFERENCES	253

LIST OF FIGURES

Figure 1.1 The Organisation of Chapters 4-7	6
Figure 2.1 Information Retrieval Procedures of a Search Engine	12
Figure 3.1 Conceptual Model	41
Figure 3.2 Notations Used in the Mathematical Model	43
Figure 3.3 Concept Indexing Framework	49
Figure 3.4 General Architecture and Main Processes	56
Figure. 4.1 Information Acquisition Framework	61
Figure 4.2 Flow Chart of Agent Controller	63
Figure 4.3 Flow Chart of a Collection Agent	64
Figure 4.4 Improved Information Acquisition Framework	68
Figure 4.5 An Example HMTL File Annotated	72
Figure 4.6 UML Class Diagram of the Improved Information Acquisition System	85
Figure 4.7 Screenshots of the Graphical User Interface	87
Figure 5.1 A Sample Entry in the Ontology Dictionary	102
Figure 5.2 The Entries Created in eWord from the Entry of the Verb “ABANDON” in WordNet 1.6	106
Figure 5.3 Mapping between eWord and OntoRo Entries	108
Figure 5.4 Illustration of the Approach based on the “One Sense Per eWord Entry” Hypothesis	111

Figure 5.5 An Algorithm for Mapping eWord and OntoRo Entries	114
Figure 5.6 Illustration of the Semantic Mapping Algorithm	117
Figure 5.7 A Converted Sentence with Ontology Tags	123
Figure 6.1 Zipf's Law	131
Figure 6.2 Normalised Zipf's Range	131
Figure 6.3 Flowcharts of the Ontology Tagging Algorithm (Training and Tagging)	135
Figure 7.1. Conceptual Model of the Knowledge Management System	151
Figure 7.2. Processing	155
Figure 7.3 Total Processing Time for Entity Indexing	159
Figure 7.4 Total Processing Time for Concept Indexing	159
Figure 7.5 Total Processing Time for Merged Indexing	160
Figure 7.6 Total Processing Time for Entity Extraction	160
Figure 7.7 CPU Processing Time for Entity Extraction	161
Figure 7.8 Memory Usage for Entity Extraction	161
Figure 7.9 Disk Usage for Entity Extraction	162
Figure 7.10 Total Processing Time for Concept Extraction	162
Figure 7.11 CPU Processing Time for Concept Extraction	163
Figure 7.12 Memory Usage for Concept Extraction	163
Figure 7.13 Disk Usage for Concept Extraction	164
Figure 7.14 Retrieving Time Comparison with MySQL Using Text after Merging of Entity and Concept Index	167

Figure 7.15 The Composition of a Document Used in the Case Study	169
Figure 7.16 Result Output and Corresponding Original Documents for Task 1	172
Figure 7.17 Result Output and Corresponding Original Documents for Task 2	176
Figure 7.18 Result Output and Corresponding Original Documents for Task 3,	179
Step 3	
Figure 7.19 Result Output and Corresponding Original Documents for Task 3,	184
Step 4	
Figure 7.20 Result Output and Corresponding Original Documents for Task 4	186
Figure 7.21 Result Output and Corresponding Original Documents for Task 5	189

LIST OF TABLES

Table 3.1 Feature Comparisons of Knowledge Representation Languages And Models	38
Table 3.2 A Simple Illustrative Example of Concept Indexing (a-k)	52
Table 4.1 Summary Results for Case Study 1	80
Table 4.2 Summary Results for Case Study 2	83
Table 5.1 Comparisons between Roget's Thesaurus and WordNet	98
Table 6.1 Statistical and Context Information	127
Table 6.2 Summary of Tests Conducted	139
Table 6.3 Average Accuracy in Case Study 1: Experiments with Different Zipf's	142
Table 6.4 Average Accuracy in Case Study 1: Experiments with Different Window	142
Table 6.5 Average Accuracy in Case Study 2: Experiments with Different Ratio S	144
Table 6.6 Average Accuracy in Case Study 2: Experiments with Different Threshold Values	144
Table 6.7 Average Accuracy in Case Study 3: Experiments with Different Zipf's	146
Table 6.8 Average Accuracy in Case Study 3: Experiments with Different Ranges Window Widths and Ratios S	146
Table 7.1 Information Types Used	156

ABBREVIATIONS

KM	Knowledge management
KMS	Knowledge management system
IR	Information retrieval
ML	Machine learning
NLP	Natural language processing
DNS	Domain name system
IP	Internet protocol
ADNS	Asynchronous DNS
LSI	Latent semantic indexing
SVD	Singular value decomposition
SVM	Support vector machine
SRM	Structural risk minimisation
NP	Non-deterministic polynomial-time
NE	Named entity
POS	Part of speech
MT	Machine translation
WSD	Word sense disambiguation
IE	Information extraction
OE	Ontology engineering
RDF	Resource description framework

RDFS	RDF Schema
KR	Knowledge representation
AI	Artificial intelligence
SW	the Semantic Web
ER	Entity relationship
DL	Description logics
UML	Unified modeling language
OWL	Web ontology language
KB	Knowledge base
HTML	Hyper text markup language
DTD	Document type definition
SD	Small dictionary
BD	Big dictionary
OT	Ontology tagging
MRD	Machine-readable dictionary
JDK	Java development kit
MI	Mutual information
UC	Untagged corpus
UBC	Untagged Brown Corpus
SQL	Structured query language
URL	Universal resource locator

CHAPTER 1. INTRODUCTION

1.1 MOTIVATION

In recent years, it has been recognised that an organisation's success is more dependent on its intellectual assets than on the value of its physical resources. Knowledge is now the key battleground for competition [Davies, 2003]. As a result, the knowledge management discipline has recently become a very active field of research. Knowledge management (KM) is defined as the tools, techniques and processes for effective and efficient management of an organisation's intellectual assets [Davies, 2000]. KM is concerned with the representation, organisation, acquisition, creation, use and evolution of knowledge in its many forms [Jurisica et al., 2004]. The KM process is an iterative process which has four stages, i.e., knowledge creation, acquisition, knowledge organisation/storage, knowledge distribution and knowledge application [Wiig, 1995]. Effective KM typically requires an appropriate combination of organisational, social and managerial initiatives along with the deployment of appropriate technologies [Marwick, 2001].

There are two forms of knowledge that are important for organisational effectiveness, namely, tacit and explicit knowledge [Nonaka, 1994]. Tacit knowledge is rooted in individual's action, experience, commitment, ideals, values and emotions. Tacit knowledge is hard to formalise because of its highly personal nature, while explicit

knowledge can be expressed using formal and systematic languages [Nonaka et al., 2000]. Therefore, explicit knowledge can be transferred between humans without human interaction. Explicit knowledge can be considered as information in the right context, i.e. information which can lead to effective action [Davies, 2000]. These two forms of knowledge are supported by technologies and tools such as collaborative working environments, speech recognition techniques, and document management systems [Marwick, 2001].

Although the concept of knowledge and knowledge management is not new, knowledge management systems (KMSs), which involve the application of IT systems and other organisational resources to manage knowledge strategically, are a relatively recent phenomenon [Quaddus and Xu, 2005]. KMSs add value to KM by providing the necessary infrastructure for organisations to implement the KM process [Dovey, 1997]. In the last two decades, advances have been made in various computing disciplines such as information retrieval, machine learning, natural language processing and ontology that set technology foundations for the further development of knowledge management systems [Marwick, 2001; Kobayashi and Takeda, 2000; Sebastiani, 2002].

Current knowledge management systems, however, have weaknesses when dealing with explicit knowledge contained in unstructured text documents, as the processes involved are normally costly and lengthy [Broder and Ciccolo, 2004; Davies, 2003; Mukherjee and Mao, 2004]. Therefore, there is a need for more efficient tools for

knowledge creation, acquisition, organisation, sharing and reuse. In particular:

1. Tools are needed to efficiently and automatically acquiring information from the web to facilitate knowledge acquisition.
2. There is a need for methodologies and tools for representing knowledge in a way that makes knowledge understandable by both humans and machines. The availability of such methodologies and tools would facilitate knowledge organisation, sharing and reuse.
3. Methodologies and tools are needed for extracting explicit knowledge from unstructured text sources, especially when the information volume becomes very large.
4. Tools for managing large volume of knowledge are also needed so that knowledge can be utilised efficiently.

These weaknesses and needs have motivated research into providing KMSs with some new capabilities. These capabilities include:

1. Automatic searching for information of interest which enables the acquisition of new knowledge into the system;
2. Use of systematic schemes for knowledge representation so that both humans and machines can understand the semantics of knowledge;

3. Automatic extraction of explicit knowledge from unstructured text information;
4. Efficient organisation of the ever-growing knowledge within the system to facilitate knowledge sharing and reuse.

Therefore, the next generation of KMSs have to be automated and able to partially understand natural languages. There are indications that information retrieval (IR), machine learning (ML), natural language processing (NLP) and ontology techniques can provide means for developing such enhanced capabilities.

1.2 AIMS AND OBJECTIVES

The scope of the research reported in this thesis is the management of explicit knowledge from a technology perspective. The overall aim is to develop ML, NLP and ontology-based techniques for knowledge management that can enable the automation of knowledge acquisition, extraction and organisation. The individual objectives of this research are:

1. To create *formal models* for content representation of unstructured text.
2. To develop *a method* for acquiring knowledge automatically from large collections of Web documents.
3. To create necessary *ontology-based resources* to facilitate knowledge extraction.

4. To create *a method* for extracting knowledge from unstructured text.
5. To develop *a conceptual model, a system architecture and a method* for knowledge management enabling efficient knowledge extraction and organisation.

1.3 OUTLINE

The main body of the thesis comprises Chapters 2 to 7. Chapter 2 is a review chapter. Chapters 3-7 address the objectives listed above. The organisation of Chapters 4-7 is shown in Figure 1.1. The final chapter, Chapter 8, summarises the contributions and conclusions of the work and makes suggestions for further research.

Chapter 2 reviews the state of the art in IR, ML, NLP and ontological engineering, and the use of these four technologies in relevant domains.

Chapter 3 addresses research objective (1). It starts with analysis of knowledge representation and its role in the knowledge management process. Then a conceptual model for knowledge representation is presented. The conceptual model is further developed into a formal mathematical model. Then, the benefits of adopting this model are outlined. Next, a framework using this model is presented with an illustrative example. Finally, a system architecture to implement this framework and the system processes are described.

Chapter 4 focuses on research objective (2). It starts with an analysis of the traditional

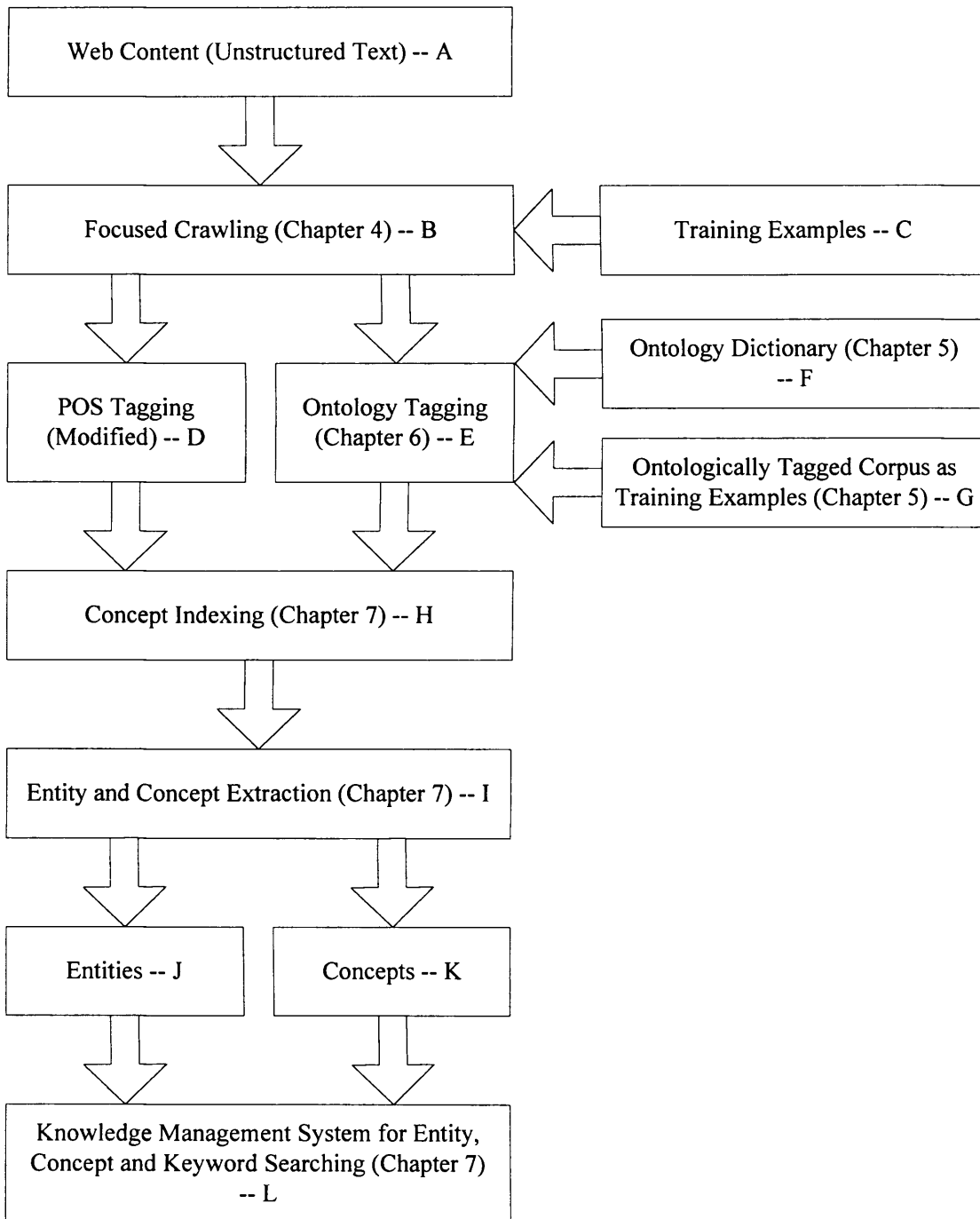


Figure 1.1 The Organisation of Chapters 4-7

knowledge acquisition methods and their weakness. This analysis shows the needs for automatic knowledge acquisition. Then, a basic method is described. The detail knowledge acquisition process is described along with detailed flow chart. Then the limitations of this basic method are analysed to highlight the improvements needed. Following that, an improved knowledge acquisition method using a supervised machine learning algorithm is described using a flow chart. Three feature selection alternatives are presented and analysed. Then the improved knowledge acquisition method for automatic knowledge acquisition based on intelligent focused crawling architecture is described. Two case studies are conducted to show the feasibility of this method in two different domains, and to test the optimisation of the machine learning algorithm to achieve better acquisition accuracy.

Chapter 5 addresses research objective (3). It starts with a discussion on ontology tagging and its benefits. Then, the resources available and those needed to achieve ontology tagging are analysed. Next, a general purpose ontology dictionary, and the developed method for semantic mapping between a lexicon and the ontology dictionary are described. Then the ontologically tagged corpus for supervised ontology tagging is developed. Tests are conducted to show the accuracy of the ontologically tagged corpus generated.

Chapter 6 focuses on research objective (4). An algorithm for full text ontology tagging using machine learning is proposed. The training and tagging processes of this algorithm are described. In order to achieve better tagging accuracy, three case studies

are conducted and evaluated to choose the best design option. Finally, the characteristics of the tagging algorithm and the factors affecting its accuracy are analysed and discussed.

Chapter 7 focuses on research objective (5). It presents the conceptual model and the system architecture for knowledge management, and the prototype system developed.

The system uses concept indexing, an existing part of speech tagging module, the ontology tagging developed in this work to index and extract entity and concept information, which are further used for entity, concept and keyword searching. Next, two sets of tests are described. The first set of the tests is conducted on indexing of entities and concepts. The purpose of the first set of tests is to examine how the processing time, extraction time and the demand of computer resources change as the volume of information/knowledge in the system grows. The second set of tests examines the retrieval performance after the merging of the entity and concept indices. The purpose of this set of tests is to evaluate the retrieving performance against an existing state-of-the-art standard solution. Finally, five case studies are conducted to illustrate the advantages of this knowledge management approach where traditional searching fail to apply or has poor performance. All examples in Chapters 3-7 used to illustrate the proposed solution are based on a prototype system developed for knowledge management.

Chapter 8 summarises the contributions made and the conclusions reached, and suggests possible directions for further investigations in this area.

CHAPTER 2. REVIEW OF TECHNOLOGY APPROACHES TO KNOWLEDGE MANAGEMENT

This chapter reviews technology approaches to knowledge management. First, knowledge management is introduced. Then knowledge management models are described. Next, four technologies for knowledge management are reviewed, which are: information retrieval, machine learning, natural language processing, the Semantic Web and ontology.

2.1 KNOWLEDGE MANAGEMENT

Knowledge now is seen at the centre of global economic transformation [Bell, 1978], and it is the most powerful engine of production organisations which are increasingly focused on management [Marsh, 1965]. Knowledge management (KM) is currently receiving considerable attention, from both academics and practitioners, and is being addressed by a broad range of academic literature and popular press.

2.1.1 Knowledge

There are many interpretations on the definition of “knowledge”. Plato [1953] first defined knowledge as “justified true belief”, which is that people believe and value on the meaningful and organised accumulation of information (messages) through experience, communication or inference [Dretske, 1981; Lave, 1988; Blacker, 1995].

The chain of knowledge flow is data-information-realisation-action/reflection-wisdom

[Kakabadse et al., 2003], where processing is conducted through each stage to transform data into wisdom.

2.1.2 Knowledge Management

There are many definitions about knowledge management. For example, in [O'Dell and Jackson, 1998], KM is defined as “conscious strategy of getting the right knowledge to the right people at the right time and helping people share and put information into action in ways that strive to improve organisational performance”; in [Beckman, 1997] KM is defined as “formalization of, and access to, experience, knowledge and expertise that create new capabilities, enable superior performance, encourage innovation and enhance customer value”. However, fundamentally, these working definitions relate to four elements: business processes, information technologies, knowledge repositories and individual behaviors [Eschenfelder et al., 1998]. These four elements enable an organisation to methodically acquire, store, access, maintain, and reuse knowledge from different sources [Eschenfelder et al., 1998]. There are generally five models of KM, viewing KM from different perspectives, which result in different approaches to KM. These models are: philosophy-based model, cognitive model, network model, community model, and quantum model [Kakabadse et al., 2003]. Cognitive model, in particular, is receiving considerable attention [Swan and Newell, 2000]. It treats knowledge as objectively defined and codified concepts and facts. It focuses on knowledge capture and storage [Kakabadse et al., 2003]. The primary aim of this model is to codify, capture and explore explicit knowledge and information, where technology

is considered as an important integrative mechanism [Kakabadse et al., 2003].

The author of this thesis supports the approach of cognitive model for KM. As the cognitive model concentrates on the use of technologies as important means to achieve KM, relevant technologies are reviewed in the following sections, which are information retrieval, machine learning, natural language processing, the Semantic Web and ontology.

2.2 INFORMATION RETRIEVAL

2.2.1 Information Retrieval and Web Search

Information retrieval (IR) studies the retrieval of information (not data) from collection of written documents. A typical IR system prepares a certain index for the given text collection and responds to queries with a list of documents ranked according to certain criteria [Chakrabarti, 2002].

A search engine is an IR system that searches Web documents for specified queries and returns a list of documents to match the queries. Typical IR procedures of a search engine are shown in Figure 2.1. During the crawling procedure, specialised agents, which are called crawlers, spiders, Web robots, or bots retrieve large quantity of Web pages simultaneously, and store them for further processing [Chakrabarti, 2002]. Preprocessing is to unify various Web pages retrieved so that they are appropriate for further processing, which normally includes Web page validation, format conversion,

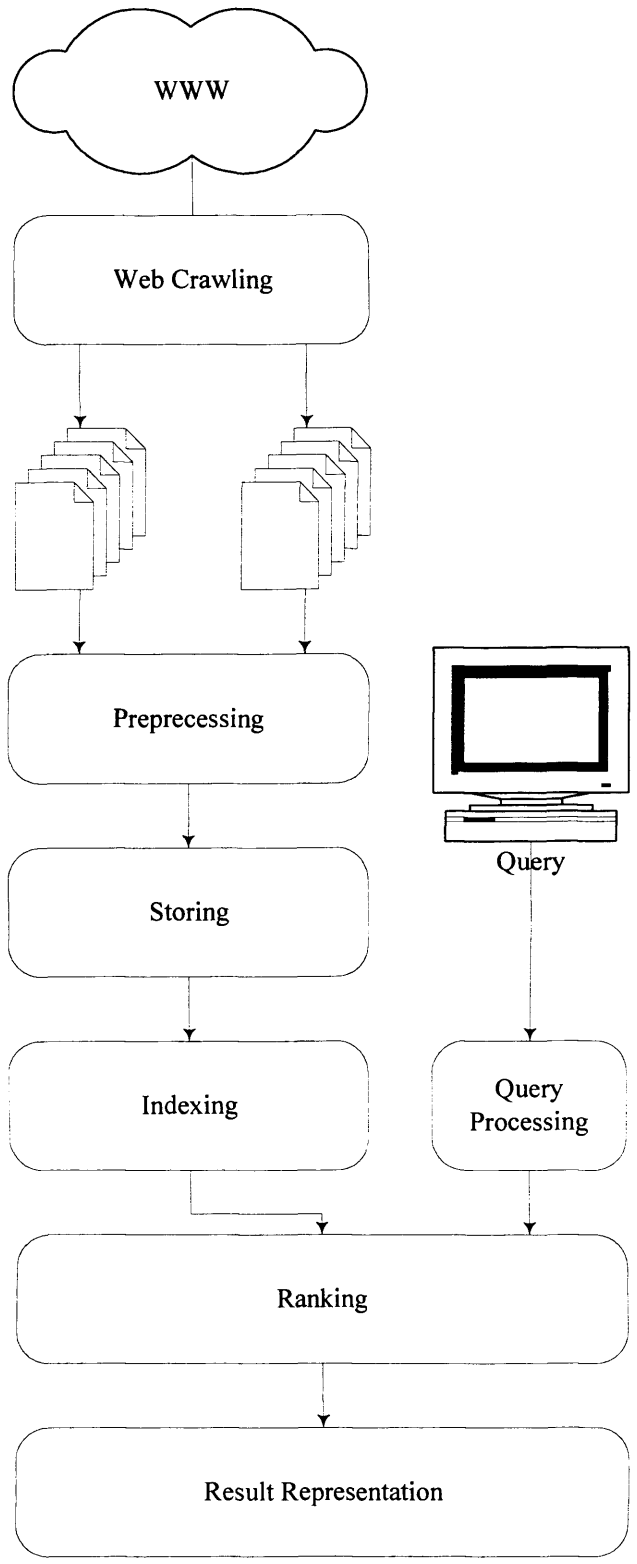


Figure 2.1 Information Retrieval Procedures of a Search Engine

stripping of stop words¹, stemming, etc. When re-formatted Web content is passed through the preprocessing procedure, the content is analysed to create an index. When a query is input to a search engine, the query is evaluated according to defined certain criteria, then the analysed query is calculated using the index and document information to rank the result set. When the ranking result is generated, it is represented to the user to answer the query.

There are several measurement metrics for IR systems, such as *precision* and *recall*, F-measure [van Rijsbergen, 1979] and mean average precision [Baeza-Yates and Ribeiro-Neto, 1999]. However, the most used measure is *precision* and *recall* [Kobayashi and Takeda, 2000]. Precision is defined as the proportion of relevant documents to all the documents retrieved:

$$P = (\text{number of relevant documents retrieved}) / (\text{number of documents retrieved});$$

and recall is defined as the proportion of relevant documents that are retrieved, out of all relevant documents available:

$$R = (\text{number of relevant documents retrieved}) / (\text{number of relevant documents})$$

2.2.2 Crawling

Effectively retrieving Web pages from the Web to a document collection is the first step for a search engine. Crawlers start the retrieving process from some given Web pages that have outbound links to other pages which have not been retrieved or refreshed.

¹ Search engines normally filter extremely common words in order to save disk space or to speed up search results. These filtered words are known as “stop words”.

New links from retrieved pages are normalised and queued for further processing. [Brin and Page, 1998] provided a first known in-depth description of a large-scale crawler as a part of early Google, being capable to process more than 24 million Web pages. A Java based large-scale crawler, Mercator, was reported in detail in [Heydon and Najork, 1999], which concentrated on the issues of scalability and the extensibility. Mercator used a specially designed data structure to handle tens of millions of Web pages with a limited size of memory. The modular design of Mercator facilitates its extension of functionalities when needed. Focused crawling was introduced in [Chakrabarti, 1999], which means only those Web pages which are classified as relevant to given topics are stored during crawling and the links in those pages are processed for future crawling. This technique is useful in discovering Web resource of interest, Web structure analysis and building high-quality collections of Web documents on specific topics [Chakrabarti, 2002].

Four factors of performance and reliability are considered important to build a large scale crawler.

1. Domain Name System (DNS) resolution issues

Domain name systems associate many types of information with domain names, but most importantly, it associates the Internet Protocol address (IP address) with a given domain name. Address resolution is one of the major bottlenecks in enabling high performance crawlers. A multithreading DNS resolver was used in [Heydon and Najork, 1999] to improve the performance of DNS resolving, and reduced the percentage of elapsed time of each thread from 87% to 25%. The asynchronous DNS (ADNS) client

library [Jackson, 2006] was used in [Chakrabarti, 2002] to accelerate the Web address resolving speed.

2. Concurrent retrieving and multithreading

Multithreading, concurrent process, non-blocking sockets with event handlers are three typical approaches to enable concurrent Web page retrieval [Chakrabarti, 2002].

Non-blocking sockets with event handlers have advantages over the other two approaches, because mutual exclusion for shared data structures and random input-output access to disk interruptions, which often happen in the other two approaches, do not occur in non-blocking sockets [Nichols et al., 1996].

3. Link extraction requirements

When hyperlinks are extracted during crawling, duplicated hyperlinks should be avoided. In addition, prohibited links by host servers should not be crawled by following the specification of Robots Exclusion Protocol². Malicious spider traps often occur in the crawling process, and sometimes cause crawlers to crash. Some traps also cause the crawler retrieve indefinite number of dummy pages, or similar pages with different depth in terms of forward slashes in hyperlinks [Heydon and Najork, 1999]. Due to this reason, crawlers need to be designed to prioritise Web links to crawl [Cho et al., 1998].

4. Refreshing of crawled pages

To keep Web pages crawled up to date, some assumptions need to be made for

² For more information refers to: <http://www.robotstxt.org/wc/robots.html>

estimating update intervals on different websites. Algorithms for refreshing crawled Web pages newer than a specified crawling period were proposed in [Brewington and Cybenko, 2000]. Incremental crawling was described in [Cho and Garcia-Molina, 2000] to refresh Web pages retrieved in a timely manner.

2.2.3 Models for Information Retrieval

An IR system needs to present documents relevant to the user's need and rank the documents retrieved in the order of predicated likelihood of relevance to the user. Different IR models are proposed based on distinct sets of premises to achieve such goals.

In general, an information retrieval model [Baeze-Yates and Riberiro-Neto, 1999] is defined as a quadruple $[D, Q, F, R(q_i, d_j)]$, where

D is a set composed of logical views (representations) for the documents in the collection;

Q is a set composed of logical views (representation) for the user's information needs (queries);

F is a method for modelling document representations, queries, and their relations;

$R(q_i, d_j)$ is a ranking function which associates a real number with a query $q_i \in Q$ and a document representation $d_j \in D$. Such ranking defines an order among the documents with regard to the query q_i .

In addition, a model could be extended to include multiple sources of evidence with both collaborative and content information [Griffith and O'Riordan, 2003]. Some

typical IR models are described as below:

1. Boolean Model

In Boolean model, documents are treated as a set of terms, and queries are expressed in Boolean expressions. The relevance of query results is calculated using set theory and Boolean algebra [Wartick, 1992]. Boolean model is the most common model in an IR system, however, from the probability distribution point view, [Verhoeff et al., 1961] proved that it is inefficient to for IR systems. Therefore, Boolean model was proposed to be used in conjunction with other IR models. For example, relevance feedback was introduced into a Boolean IR system to provide the precision of a Boolean search and the advantages of a ranked output in [Radecki, 1982]. A Boolean model was also used in an IR system to rank documents by exploiting term dependence information from a thesaurus to achieve higher retrieval effectiveness than some of the previous methods proposed [Lee et al., 1993].

2. Vector Model

Vector model was made popular by Salton and the SMART system [Salton and Lesk, 1968; Salton, 1971; Salton and Buckley, 1988]. Salton et al. [1982] introduced the extended Boolean model to exploit benefits from both Boolean model and vector space model, and tests indicated the system produced better results than either of the model used alone. A sense-based vector space retrieval model was presented in [Stokoe et al., 2003], which improved the precision by 45.9% relatively to traditional TF*IDF³ term

³ The tf-idf weight (term frequency–inverse document frequency) is a weight often used in information retrieval modelling

weighting techniques. Latent Semantic Indexing (LSI) was proposed in [Furnas et al., 1988] to reduce the dimensions of the vector space by using low-rank approximation based on singular value decomposition (SVD). Karypis and Han [2000] proposed a fast dimensionality reduction algorithm with lower computational requirements and the ability in supervised learning, compared with LSI.

3. Probabilistic Model

Probabilistic model is an IR model based on a probabilistic interpretation of document relevance to a given user query [Baeza-Yates and Ribeiro-Neto, 1999]. There are two major types of probabilistic models: relevance models and logic inference models [Fuhr, 1992]. Generally, probabilistic models reveal better performance than Boolean models [Crestani et al., 1998]. Cooper [1994] raised questions on using probabilistic ranking in information retrieval, who stated that the cost of creating and trouble shooting probabilistic IR theories is high.

4. Fuzzy Set Model

Radecki [1976] described using fuzzy set theory [Zadeh, 1965] in information retrieval to enable different degrees of importance of particular terms in search patterns. In [Miyamoto et al., 1983] fuzzy set theory was used to produce a pseudo-thesaurus for IR to enable searching for different but related keywords in the documents. Wong and Yao [Wong and Yao, 1995] proposed a unified framework of probability inference which provided conceptual and mathematical basis for models such as fuzzy set.

5. Other Models

Neural network model was used in an IR system in [Wilkinson and Hingston, 1991],

and the test results showed that many standard search strategies are applicable in the neural network model, such as cosine vector measure method. A probabilistic IR model based on feed forward artificial neural network was implemented in [Kwok, 1995]. Turtle et. al. introduced inference networks for IR in [Turtle et al., 1990]. Belief network model was introduced in [Ribeiro-Neto and Muntz, 1996] to generalise the inference network model [Turtle et al., 1990] and other classical probability based models to improve the retrieval performance.

2.2.4 Indexing

The purpose of indexing is to reduce the retrieval time in an IR system which contains large document collections, and it is especially useful for large and slow growing text collection [Baeze-Yates and Riberiro-Neto, 1999].

An inverted index is an index structure storing a mapping from words to their locations in a document or a set of documents [Harman et al., 1992]. To improve the search speed, inverted files need to be appropriately compressed and encoded [Witten et al., 1999]. Eleven indexing compression models were described and compared in [Witten et al., 1999]. There are performance trade-offs between these eleven models, such as compression rate, hard drive usage, and memory usage. Therefore, the appropriate model chosen depends on various factors such as the user requirements and hardware availability [Witten et al., 1999]. A hybrid approach of indexing, called two-level searching, was developed in Glimpse system [Manber and Wu, 1993]. This approach combined a small index and sequential search techniques to reduce the size of the

inverted index and to support approximate matching [Araujo et al., 1997].

2.2.5 Storage and Query

Text compression and pattern matching are also important in IR systems. Text needs to be compressed to significantly reduce the storage size and improve the retrieving speed.

Huffman coding was first presented in [Huffman, 1952], which is an entropy encoding algorithm for lossless data compression. An efficient compression and decompression scheme for word-based Huffman encoded text file was described in [Moura et al., 1998]. Huffman encoding was used in MG system for text compression due to this method's relatively fast encoding and decoding speed and random access ability [Witten et al., 1999].

Pattern matching is widely used during the process of indexing and query. An algorithm using bit-parallelism to support extended patterns allowing errors was presented in [Wu and Manber, 1992] and an online searching tool, Agrep, based on this algorithm was developed in [Wu and Manber, 1992]. The fastest bit-parallel approximate pattern matching algorithm implemented in [Baeza-Yates and Navarro, 1999], was based on simulating non-deterministic finite-state automata [Hopcroft et al., 2000].

2.2.6 Applications of Information Retrieval

Various applications for IR were reported in literature. WebGlimpse, a tool combining searching and browsing through a neighbourhood search, was described in [Manber et al., 1997]. An IR system that included the addition of concepts to facilitate the

identification of the correct word sense was presented in [Henstock et al., 2001]. A filtering system SIFTER, which was based on a model using multiple adaptation techniques to cope with uncertainties, demonstrated a good performance in filtering documents in a realistic setting [Mostafa et al., 1997]. An IR system using semantic annotation and semantic relations in medical domain was presented in [Vintar et al., 2003]. In this system, discovery of new relation instances improved the retrieval performance.

2.3 MACHINE LEARNING TECHNIQUES

Machine learning (ML) techniques such as neural networks, genetic algorithm, simulate annealing [Pham and Karaboga, 2000; Pham and Xing, 1995] and fuzzy logic [Zadeh, 1994] in the soft computing paradigm have been used in clustering, filtering, information extraction, information personalisation, knowledge discovery and other natural language processing applications [Mooney, 2003]. Those techniques are mainly used to investigate three types of Web information, which are content, usage, and link structure [Baeza-Yates, 2003]. Most ML methods concern the task of categorising examples described by a set of features [Mooney, 2003]. In particular, three machine learning techniques relevant to the scope of this thesis are reviewed below, which are support vector machine, rule-based learning and memory-based learning techniques.

2.3.1 Support Vector Machine

Support Vector Machine (SVM) is a statistical, supervised learning method based on

structural risk minimisation (SRM) and kernel functions [Boser et al., 1992]. One major advantage of SVM is the high classification accuracy given few training examples and vague distinctive margin between different groups of examples, which was theoretically proven in [Vapnik, 1999; Scholkopf and Smola, 2002]. In practice, [Joachims, 1998] discussed algorithmic and computational costs for managing large training tasks in the SVM^{light}, an SVM implementation in C programming language. LIBSVM [Chang and Lin, 2002] was provided as an SVM library package by Chang et al.

In application, Joachims [2001] proposed a learning model for text classification based on SVM. In [Li and Liu, 2001], a Chinese web page classifier using SVM and unsupervised clustering was implemented, which mitigated the high training cost using an unsupervised clustering method. A method which combines evidence from a document and citing documents by using SVM and entropy based feature extraction was presented in [Glover et al., 2002] to improve the web page classification accuracy. In [Chen and Dumais, 2000], a user interface which automatically categorising search results using SVM text classification shows 50% improvement in finding information in search results.

2.3.2 Rule-Based Learning

The learned knowledge in rule based learning is represented in a declarative, symbolic form of logical rules as opposed to a numerical model obtained by using SVM. One advantage of this method over statistical ML methods is that the acquired knowledge is

represented in a symbolic form that can be more easily interpreted, modified and maintained by humans. Rules are normally induced from a set of training examples using a variety of algorithms [Mitchell 1997, Langley 1996]. Although the optimal goal is to construct the smallest rule set to consistently represent the training data, but normally this is a non-deterministic polynomial-time hard (NP hard) problem, which may take unacceptable long time to solve. Therefore, heuristic approaches are often used to create the approximately optimal rule set.

Rule-based learning algorithms have been applied to many areas, such as named entity (NE) recognition [Isozaki, 2001], language engineering⁴ [Bontcheva et al., 2004], and knowledge management [Davies et al., 2005].

2.3.3 Memory-Based Learning

Different from rule-based learning methods, memory-based learning (also called case-based or instance-based methods) do not generate abstract models from given examples, but rather compare the similarity of the new input examples with existing correct examples using some similarity comparison metrics. Some typical metrics are Hamming distance and Euclidian distance [Mooney, 2003]. The advantage of this method applied in NLP is that it does not discard valuable low frequency information, which often occurs in natural language. Benchmark tasks in areas from phonetics to semantics in natural language processing were conducted using memory-based

⁴ Language engineering is the creation of natural language processing systems whose cost and outputs are measurable and predictable.

learning methods with good results obtained, compared with other ML methods used [Daelemans, 2005]. Tilburg Memory-Based Learner (TiMBL), a memory-based learning software package, was implemented for language engineering purpose in [Daelemans, 2004].

2.4 NATURAL LANGUAGE PROCESSING

Natural language processing (NLP) is a field of information science to research how to enable computers to process and understand human languages. It has tight relation with information retrieval and knowledge management. Five areas in natural language processing where are related to this thesis are reviewed.

2.4.1 Part of Speech Tagging

Part of speech (POS) tagging, which is also called word-class tagging, or grammatical tagging, is to assign parts of speech (such as noun, verb, adverb, adjective) to words in a text. POS tagging is used in many other NLP areas such as machine translation (MT), word sense disambiguation (WSD) and information extraction (IE) [Voutilainen, 2003]. Current POS taggers have high tagging accuracy (about 97%). A rule-based POS tagger developed by Brill, achieved 97% tagging accuracy [Brill, 1992]. Its compact structure has an advantage over statistical POS taggers. Second-order hidden Markov model for POS tagging with the tagging accuracy (96.9%) was developed in [Thede and Harper, 1999]. Both probabilistic and rule-based tagging modules were used in tagger CLAWS4 [Garside et al., 1997] to tag British National Corpus, with a tagging accuracy

97% across the whole corpus.

2.4.2 Word Sense Disambiguation

Word sense disambiguation (WSD) is the process of identifying the meaning of words in context [Stevenson and Wilks, 2003]. WSD can be applied to machine translation (MT), information extraction (IE), information retrieval (IR) etc. Dictionary-based and machine learning are two main approaches to WSD [Stevenson, 2003], where machine-readable dictionary (MRD) (such as WordNet [Fellbaum, 1998]), semantically tagged corpora (such as Semcor package in WordNet software) and thesaurus (such as Roget's Thesaurus [Roget, 2003]) are three main knowledge sources used in both approaches [Yarowsky, 1992].

Studies show that IR systems may substantially benefit from using WSD techniques. As reported in [Stokeo et al., 2003], a disambiguation rate above 60% can improve the precision and recall of information retrieval. A, WordNet [Fellbaum, 1998], was used with a WSD algorithm to improve the precision and recall of a keyword-based IR system [Mihalcea and Moldovan, 2000].

2.4.3 Concordance and Collocations

A concordance is a list showing all the occurrences and contexts of a given word or phrase, which are found in a corpus; collocations are groups of words which frequently appear in the same context. Concordance and collocations are useful tools for tasks such as corpus annotation and WSD. For example, Yarowsky [1993] observed that,

sense is usually consistent in one discourse and nearby words provide strong and consistent clues to the sense of the target word, which is called “One Sense per Discourse”. Yarowsky [1993] also claimed “One Sense per Collocation” which states that with a high probability an ambiguous word has only one sense in a given collocation. These two claims were used in an unsupervised learning algorithm for WSD purpose, which achieved the accuracy between 90% and 96%.

2.4.4 Concept Indexing

There are different definitions of “concept indexing” in different research areas. For example, a fast dimensionality reduction algorithm called “concept indexing” [Karypis and Han, 2000] uses a clustering technique based on mathematics in a vector space model. While in [Voss et al., 1999] concept indexing refers to the process of marking interconnected concepts to their occurrences in the text collection using hyperlinks. The process is produced by a team of people to manually mark concepts in documents. Similar term “semantic indexing” in [Chang and Schatz, 1999] refers to a statistical similarity method to capture relationships between concepts and form a concept space to suggest alternative terms semantically related to query terms. This approach requires the use of a cluster of high performance computers. In [Mihalcea and Moldovan, 2000], “semantic indexing” refers to the usage of word senses in the process of document indexing, where word-based and sense-based indexing are combined to improve the precision and recall of an IR system. Holub in a recent work [Holub, 2003] introduced an iterative clustering method called “conceptual document indexing” which extracts

significant topical concepts from clustered documents and the hierarchical relationships between them, such that documents are organised in a hierarchy for browsing purpose.

2.4.5 Information Extraction

Information extraction (IE) refers to the automatic identification of selected types of entities, relations, or events in free text [Grishman, 2003]. There are many potential application areas for IE: situations where information is being extracted manually, but at a very high cost and low speed; situations where unstructured information needs to be structured; situations where business, political, military events on particular subjects need to be monitored etc. The general approach for IE is through machine learning techniques to generate extraction rules or statistical models automatically from annotated text corpora [Grishman, 2003].

A rule-based entity recognition system, MACE, was developed in [Maynard et al., 2003] to extract named entities, which achieves comparative performance with other systems based on statistics. ANNIE [The University of Sheffield, 2006] is an Information Extraction system developed using GATE, and it can extract entities, such as person, location, organization, date, and address from Web pages. A knowledge management system using information extraction techniques for automobile manufacturing intellectual properties was developed in [Hou et al., 2005] to save the labour cost of knowledge engineers and domain experts.

The state-of-art entity recognition methods have achieved relatively good results, but

for tasks such as event extraction, relation extraction results are still far from satisfaction [Stevenson, M., 2004]. Furthermore, it is time-consuming to produce rules for rule-based systems and large volumes of annotated training text is still needed for statistics-based IE systems [Maynard et al., 2003].

2.5 THE SEMANTIC WEB AND ONTOLOGY

The Semantic Web [Berners-Lee et al., 2001] is an ongoing initiative to extend the current Web structure, and standardise the descriptions of available resources on the Web, so that the information is understandable by machines, which could make the information processing more automated and less intervened by humans.

2.5.1 Ontology Engineering

Ontology engineering (OE) is a key enabling technology for the Semantic Web [Davies et al., 2005], which allows explicitly specifying concepts and their relationships in a domain in a formal way [Corcho and Gomez-Perez, 2000]. As there are many different language specifications for OE, such as DAML+OIL [Horrocks et al., 2002], SHOE [Luke et al., 1997], RDF Schema [Brickley and Guha, 2004], OE tools are needed to allow users to concentrate on modelling at conceptual level while separating from low-level syntax and specifications of different languages for the Semantic Web [Corcho et al., 2003]. Protégé is such a graphical tool commonly used for ontology editing and knowledge acquisition [Noy et al., 2001].

Currently OE tools and methodologies have been applied to applications to mitigate the

difficulties in knowledge acquisition. Maedche et al. proposed a general OE framework for semi-automatic acquisition of both taxonomy and non-taxonomy conceptual relation discovery [Maedche and Staab, 2000]. In this framework, an algorithm based on generalised association rule was used to detect relations and to determine the appropriate level of abstraction to define relations. [Gottgroy et al., 2003] described an OE approach to discover knowledge from data in evolving domains such as biological sciences, medical sciences, and social sciences.

2.5.2 Information Extraction for the Semantic Web

Stevenson et al. [Stevenson and Ciravegna, 2003] pointed out the existing gaps between current Web and the Semantic Web, and practical adoption problems for the Semantic Web. The extension from current Web to the Semantic Web requires high cost effort, which hinders the likeness of this dramatic change in the real world in the immediate future [Fensel, 2002; Avello et al., 2002]. By using information extraction, this gap could be closed [Stevenson and Ciravegna, 2003].

A formal information extraction framework for today's non-semantic web to extract knowledge was presented in [Arjona et al., 2003]. It associates semantics with the information extracted by developing knowledge channels. Information extraction has been effectively applied in some restricted domains. Jacobs and Rau, developed SCISOR, a prototype system that extracts information from financial news to find and summarize corporate merger stories [Jacobs and Rau, 1990]. Amilcare, a system for adaptive information extraction for the Semantic Web in the hope to automate or

semi-automate the information annotation process was described in [Ciravegna and Wilks, 2003]. KIM, a platform for semantic indexing, annotation and retrieval, was developed towards information extraction for the Semantic Web [Popov et al., 2003]. This platform is built based on GATE, a general architecture for text language engineering [Cunningham et al., 2002] and Semantic Web compliant knowledge representation and management.

2.6 DISCUSSION

The review of four technologies for knowledge management shows that the trend moving from information-centred society towards knowledge-centred society is emerging. These four technologies are now moving closer to support and complement each other to facilitate process of transforming data into knowledge in the chain of knowledge flow.

Section 2.2 has reviewed the area of IR. Despite of the progress made in IR research, users are still unsatisfied with the performance of the current IR systems, due to the reasons such as the slow retrieval speed, communication delays, and poor quality of retrieved results [Kobayashi and Takeda, 2000]. On one hand, users are suffering from information overload, and on the other hand, users still cannot efficiently find the information they want. Researchers in IR field found that the ambiguity in natural languages hinders the performance improvement in IR systems. This is one of the main reasons why IR community has increasing interest in the research of natural language

processing, such that by using NLP techniques could improve the current performance of IR system [Stokeo et al., 2003]. Furthermore, due to the “knowledge acquisition bottleneck” [Boicu, 2001], more efficient automated methods to acquire high quality information are needed to assist knowledge management. In this case, the expertise from IR field for exploring vast amount of online information can contribute to this growing need, where new techniques such as focused crawling, concept indexing emerge.

Section 2.3 has reviewed three ML techniques which could be potentially used in IR, NLP and KM. The research field of IR has a long history using ML techniques. NLP needs the assistance from ML to automate its processes due to vast text-based data involved. ML is being adopted in IE for building the Semantic Web to reduce the high building cost involved. Therefore, there is a trend towards combining ML into improved formal models of IR, NLP and KM [Kobayashi and Takeda, 2000; Allan et al., 2003; Baeza-Yates, 2003].

Section 2.4 has reviewed five aspects of natural language processing. Tagging techniques and semantic analysis have received increasing attention in the KM, Web mining and text mining research areas. However, there is no one system has addressed the use of concept-based tagging techniques in KM in a systematical way. Therefore, there is a potential opportunity for exploring the combination of these techniques further into a KM system.

Section 2.5 has reviewed techniques and tools for ontology building, and the building of the Semantic Web. For building the Semantic Web, processes need to be automated

because of the vast Web content information involved. Also due to the fact that most online information is based on natural languages, NLP is therefore a key technology to enable the transformation from current Web structure to the Semantic Web.

Therefore, research opportunities are open to combine these four technologies in a systematical way to streamline the knowledge flow for embracing the trend towards knowledge-centred societies. In particular, techniques such as focused crawling, machine learning, concept indexing and ontology are worth being exploited systematically in Chapters 3-7 for knowledge management.

2.7 SUMMARY

This chapter has outlined the research that addresses knowledge management, and its four enabling technologies, which are information retrieval, machine learning, natural language processing, the Semantic Web and ontology. Important trends within the scope of this work have been discussed and summarised to provide starting points for the research presented in Chapters 3-7.

CHAPTER 3. FORMAL REPRESENTATION OF CONTENT USING CONCEPT INDEXING

This chapter addresses the first objective of this research. It introduces the idea of concept indexing and presents the two models developed: conceptual and mathematical. A concept indexing framework is then introduced. The technical approach proposed is clarified using an illustrative example. Finally, the chapter describes the system architecture developed for this framework.

3.1 CONCEPT INDEXING

An analysis of the conventional approaches to knowledge representation is a starting point in the development of a new knowledge representation scheme, which uses concept indexing.

3.1.1 Conventional Models for Knowledge Representation

Knowledge representation (KR) is the study of how knowledge about the world can be represented and how reasoning can be conducted based on this knowledge. In artificial intelligence (AI), KR can play five roles: “a surrogate, a set of ontological commitment, a fragmentary theory of intelligent reasoning, medium for efficient computation and medium of human expression” [Davis et al., 1993]. In the management of explicit knowledge, KR is tightly related to other processes in knowledge management as KR can affect the way knowledge is processed and utilised [Davis, 1993]. There are a

number of modelling languages and formalisms for (KR) developed in different domains, ranging from databases to the Semantic Web (SW). Examples of such include semantic networks [Quillian, 1968], frame-based knowledge modelling [Minsky, 1975], Entity-Relationship (ER) modelling [Chen, 1976], description logics (DL) [Brachman and Schmolze, 1985], UML [Booch et al, 1998], RDF and RDFS [Klyne and Carroll, 2004], and OWL [McGuinness and van Harmelen, 2004].

- 1. Semantic networks.** A semantic network is a model for representing knowledge in patterns of interconnected nodes and arcs.
- 2. Frame-based knowledge modelling.** In frame-based knowledge modelling, collections of related frames are linked together into frame systems to represent knowledge, where the frame is a data structure for representing stereotyped situation knowledge.
- 3. Entity-relationship (ER) modelling.** ER is a data model for multilevel descriptions of views of data. ER is typically used to design the data organisation within databases or information systems. An entity is an object or concept where data is stored. A relationship is the connection through which data is shared between entities.
- 4. Description logics.** Description logics are knowledge representation languages tailored for expressing knowledge about concepts and concept hierarchies, which are sub-languages of predicate logic to provide reasoning support.

5. **UML.** Unified Modelling Language (UML) is a non-proprietary, third generation modelling and specification language, which has been applied into different areas such as software engineering, designing complex engineering systems, business process, and organisational structures.
6. **RDF and RDFS.** Resource Description Framework (RDF) is a modelling language to describe resources on the web in a domain-independent way. Users can define their own terminology in a scheme language called RDF Schema (RDFS).
7. **OWL.** The Web Ontology Language (OWL), which is built to enhance RDF, is an ontology language for writing explicit, formal conceptualisations of domain models.

The existing KR languages and formalisms will be analysed using the requirements defined below to decide whether these languages and formalisms are suitable for KM.

3.1.2 Requirements for Knowledge Representation of Explicit Knowledge

The requirements for representing explicit knowledge for the needs of knowledge management are summarised below.

1. **Automatic extraction of knowledge from unstructured text.** Automating knowledge extraction can dramatically reduce the labour cost as opposed to

extracting knowledge manually. Therefore, automatic knowledge extraction is an essential feature in knowledge representation.

2. **Support for efficient reasoning.** One of the reasons for developing more advanced knowledge representation schemes is the need for reasoning support. Knowledge representation is needed to support several reasoning capabilities such as reasoning on class membership, equivalence and classification reasoning [Antoniou and van Harmelen, 2004].
3. **Rich lexical and semantic representation.** Explicit knowledge is normally expressed in unstructured text-based documents. Examples of this are emails, web pages, electronic documents, and minutes of meetings. Documents differ from data, in that data does not have lexical or semantic information, whereas documents contain both lexical and semantic information. The support for syntactical and semantic representation of such documents is an important requirement. Necessary syntactical and semantic information should be efficiently extracted and maintained.
4. **Good scalability to large document collections.** One of the problems with knowledge management systems of organisations is the ever growing size of their text-based documents. The large size of these documents sometimes increases the processing time considerably, which results in delayed delivery of the knowledge required.. Therefore, good scalability is required for large document collections.

5. Flexible ways of making queries. Flexible ways mean that users can input queries for information and knowledge at different levels of abstraction. In other words, users may use *concrete instances of concepts* (i.e. entities) or only input *concepts*, when they do not have concrete realisations of those concepts or when they do not need instantiations of any concepts to make queries [Chakrabarti, 2002]. For example, when a user is sure about what he/she wants to find out about a “Volvo” car, then he/she will probably input “Volvo” as a key query term for the information. If the user wants only information/knowledge at a more abstract level i.e. car industry, he/she will probably input “car” instead of many different car brands at the same time. At this time, the user treats “car” as a *concept* with a more abstract level than “Volvo”. Therefore, there is a need for more flexibility in making queries.

The knowledge representation languages and formalisms outlined in this section are further analysed through an extended literature review including [Quillian, 1968; Minsky, 1975; Chen, 1976; Brachman and Schmolze, 1985; Booch et al, 1998; Klyne and Carroll, 2004; McGuinness and van Harmelen, 2004], to investigate how and to what degree they support these five requirements. In Table 3.1, “good”, “fair”, “poor” and “difficult” indicate different levels of support for each requirement. “Unknown” is used when there is no information available for the corresponding requirement.

Table 3.1 Feature Comparisons of Knowledge Representation Languages And Models

Modelling languages and formalisms for knowledge representation	Requirements				
	Automatic extraction of knowledge from unstructured text	Support for efficient reasoning	Rich lexical and semantic representation	Good scalability to large document collections	Flexible ways of making queries
Semantic Networks	Not easy	Poor	Fair	Good	Poor
Frame-Based Knowledge Modelling	Not easy	Fair	Fair	Poor	Poor
Entity-Relationship (ER) Modelling	Difficult	Poor	Poor	Good	Poor
Description Logics	Difficult	Fair	Poor	Poor	Good, but unintuitive
UML	Difficult	Unknown	Poor	Poor	Difficult
RDF And RDFS	Difficult	Good	Poor	Good	Fair, but unintuitive
OWL	Difficult	Fair	Poor	Good	Good, but unintuitive

As Table 3.1 shows, there is still a lack of support for automated extraction of knowledge from unstructured text. Also, there is a lack of support for syntactic and semantic information representation. Moreover, current modelling languages and formalisms are not suitable for making queries, because their representation schemes are rigid, complicated and not intuitive. Also, because of the trade-off between the expressiveness of KR and the effectiveness of logic reasoning [Antoniou, 2004], a *proper* level of expressiveness of KR to support *reasonable* speed of reasoning is needed. To meet these requirements, the idea of concept indexing is introduced, defined and formalised using a concept indexing framework

3.1.3 Definitions

In the context of this thesis, an **entity** is an identifiable and discrete instance existing in a text document. A **concept** is the abstract or physical information about entities or the relationships between them. An **index** is “the systematic guide to items contained in a collection of *concepts*” [ANSI, 1984]. **Concept index** is a machine understandable index of entities and concepts in document collections. In the context of knowledge management, **concept indexing** could be defined as the analytic process of identifying the entities and relationships which represent the knowledge conveyed in a document in the form of a **concept index**.

An assumption made here is that the information conveyed in a text can be analysed separately in terms of the entities and concepts contained. This work suggests to extract entities from unstructured text-based content using a *language knowledge base* (KB),

and to identify concepts with the help of a *concept KB*. Once entities and concepts are isolated, they can be used to build a concept index.

The conceptual model of concept indexing is illustrated in Figure 3.1. As the figure shows, the knowledge from unstructured text-based content is analysed and structured using entities and concepts with the help of two knowledge bases, the language KB and the concept KB.

The **Language KB** contains the lexical and grammar knowledge. Lexical knowledge provides a background for part of speech (POS) analysis and further language analysis. Grammar knowledge is used to determine whether the compositions will be treated as entities or concepts in this language analysis.

The **Concept KB** contains ontology knowledge for a large vocabulary of words and phrases, so that the system can “understand” the meaning of the text. For example, if the words “cars” and “vehicles” are frequently used in a text, then the system “knows” that the text contains some information about “transportation”, which is a more abstract concept than “cars” and “vehicles”.

Furthermore, the knowledge organised in the form of entities and concepts is stored using a concept index. The concept index is essential because, as the knowledge base in the system grows larger, finding entities and concepts efficiently from the knowledge base will be increasingly difficult. With no efficient methods for processing, storing and retrieving, the knowledge base will finally become unmanageable and of not much value.

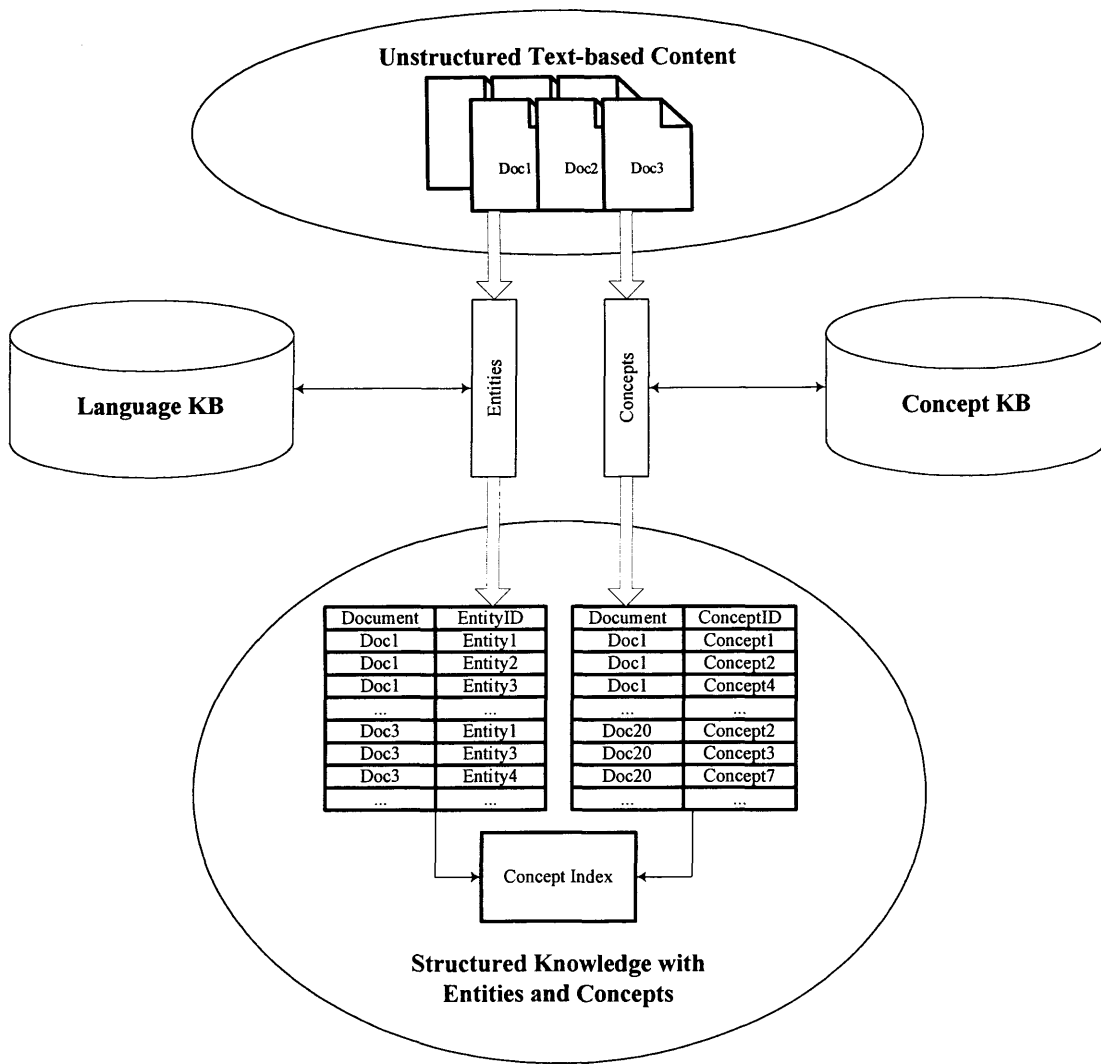


Figure 3.1 Conceptual Model

3.1.4 Mathematical Model

The concept indexing model proposed is further developed using a mathematical model.

The notations used are illustrated in Figure 3.2.

Given a text Ψ from a text collection Σ , the knowledge in this text is defined as Ξ , which can be decomposed into entity knowledge set Γ and concept knowledge set E .

Entity knowledge in Σ is defined as M , and concept knowledge is defined as N . All knowledge in Σ is defined as K . An element of the entity knowledge set Γ is defined as φ , and an element of the concept knowledge set E is defined as ε . i and j are used as subscription indexes to differentiate different elements in the entity knowledge set and concept knowledge set. The mathematical model is defined as follows.

IF	
$\Gamma \subseteq M$	(3.1)
AND	
$E \subseteq N$	(3.2)
AND	
$\Xi = \Gamma \cup E$	(3.3)
AND	
$K = M \cup N$	(3.4)
THEN	
$\Gamma = \{\forall \varphi_i \subset \Gamma, \forall \varphi_j \subset \Gamma, \varphi_i \neq \varphi_j \mid \bigcup_{i=1}^n \varphi_i = \Gamma, \varphi_i \cap \varphi_j = \emptyset, \exists \varepsilon_k \subset \varphi_i\}$	(3.5)
AND	
$E = \{\varepsilon_i, \varepsilon_j \mid \bigcup_{i=1}^n \varepsilon_i = E, \varepsilon_i \neq \varepsilon_j\}$	(3.6)

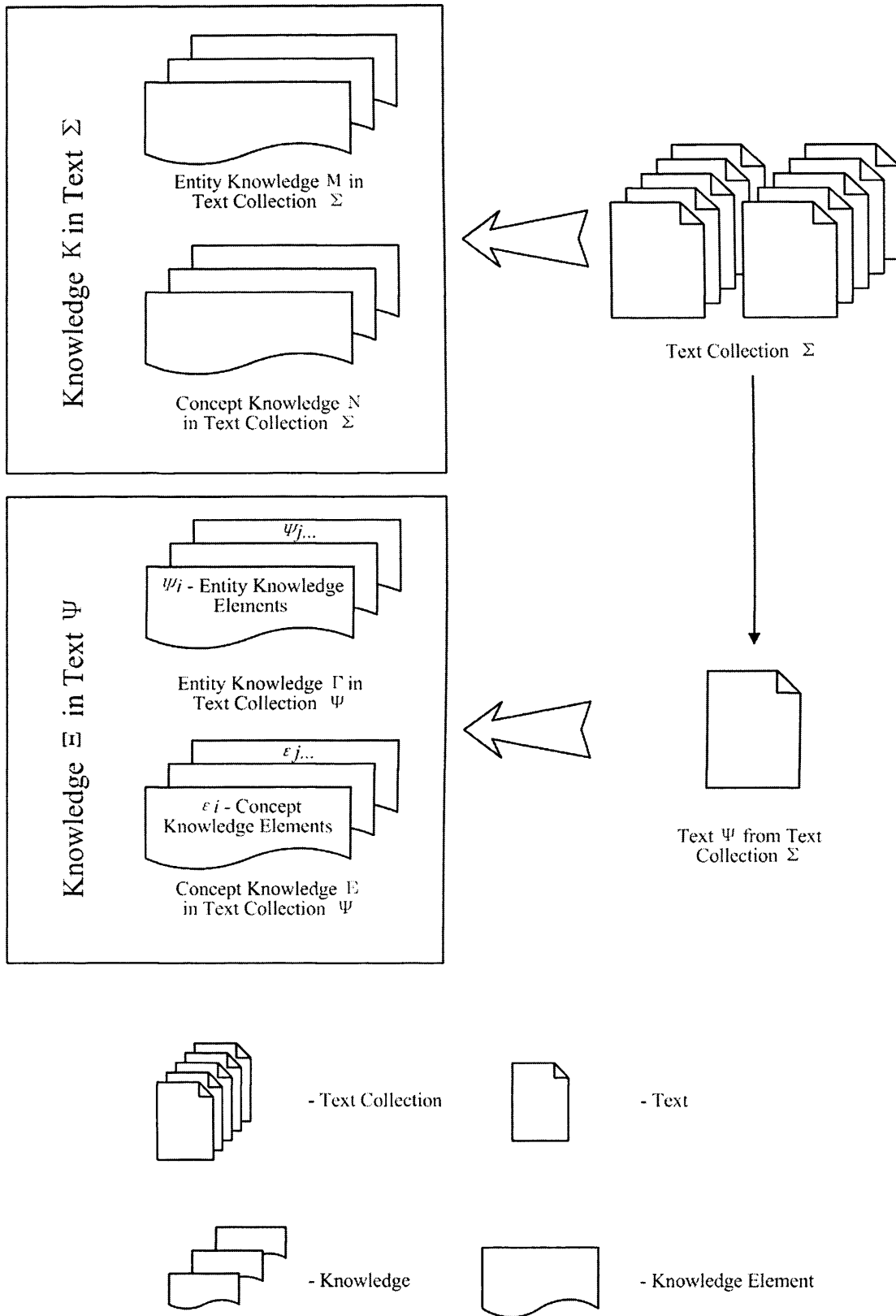


Figure 3.2 Notations Used in the Mathematical Model

(3.1) Assumes that entity knowledge Γ in a text Ψ is a subset of the entity knowledge M in the text collection Σ .

(3.2) Assumes that concept knowledge E in a text Ψ is a subset of the concept knowledge N in the text collection Σ .

(3.3) Assumes that information Ξ in a section of text Ψ is the union of entity knowledge set Γ and concept knowledge set E in the text Ψ .

(3.4) Assumes that information in the text collection Σ is the union of entity knowledge set M and concept knowledge set N .

(3.5) When (3.1)-(3.4) are satisfied, then entity knowledge Γ from text Ψ is a union of entity knowledge elements φ , and entity knowledge elements do not have intersections with each other, and there exists a concept knowledge element ε which is a subset of an entity knowledge element, but not always.

(3.6) When (3.1)-(3.4) are satisfied, the concept knowledge Γ from text Ψ is a union of concept knowledge elements ε . They may have intersections.

This approach to defining knowledge contained in a text collection has some substantial benefits. First of all, entities, which are normally easy to express using keywords, can be separated from concepts, which are normally difficult to express with keywords. For example, it is easy to specify entities about a specific company, but it is difficult to specify information about a product of a specific company which may introduce negative effects on people unless the user knows the specific name of the negative

effect such as “cancer”, and this is because information related to a specific entity is easier to express than an abstract concept conveyed by this information. From (3.5) and (3.6), the information Ξ in text Ψ from text collection Σ is formed as a union entity information elements φ and concept information elements ε . This separation of information, which is normally processed as keywords in an IR system, makes the knowledge representation clearer when considering different levels of abstraction of the information contained in a knowledge management system. It also facilitates the queries for information from the knowledge management system. In this sense, concept information is expressed in a more appropriate way as the high level queries are not “overfitted” by unnecessarily detailed keywords, as using keywords to express abstract concepts tends to add more information than the information that the concepts really carry.

3.1.5 Benefits of Adopting Concept Indexing

The benefits of adopting concept indexing are discussed in this section in relation to the requirements defined in section 3.1.2.

First of all, concept indexing enables **automatic knowledge extraction**. There are two types of knowledge extracted from the text content, fact knowledge and abstract semantic knowledge. The fact knowledge is expressed by using entities and concepts whereas the semantic knowledge is expressed using POS and concepts tags. For example, given a sentence “The Fulton County Grand Jury said Friday an investigation of Atlanta’s recent primary election produced no evidence that any irregularities took

place”, “The Fulton County Grand Jury” as an entity, “law” is an abstract concept for “jury”, and “election” will be treated as fact knowledge.

Secondly, this approach enables **efficient reasoning** using membership, equivalence and classification. As concepts tags are attached to all meaningful words/phrases in the text, the membership reasoning can be deduced according to the concept hierarchy in the concept KB. Equivalence can be conducted at different abstraction levels, as well as the instance level as exact term matching. With the help of the current state-of-art keyword-based indexing technique, classifications and equivalence reasoning will be fast; the ability to perform reasoning in batch mode also improves the efficiency.

Thirdly, concept indexing enables **rich lexical and semantic representation**, and the use of other sets of symbols in the future, if needed. Concept index enables rich semantic representation, in that POS information and ontology information are treated as two distinct sets of symbols, similar to the way the keyword-based models treat words and phrases. Therefore, as long as the meanings of the symbols are defined, the same indexing mechanism can be applied to these sets of symbols, no matter how many sets of different symbols are employed in the system. For example, if in the future, phonetic information is needed, a set of phonetic symbols can be defined and added into the system. Therefore, this structure gives more expressive power than previous models, which do not have lexical and semantic representation capabilities.

Next, this approach can be applied to **large document collections**. Because the concept indexing model is built using existing keyword-based indexing technologies, large

document collections will not be a problem, as current keyword-based indexing tools have been proven to handle large volumes of data efficiently.

Furthermore, this framework enables more **flexible ways of conducting queries** compared to knowledge base systems that use traditional keyword-based queries. The availability of lexical and semantic information, POS symbols, ontology symbols and keyword symbols, which can be input to the system as queries at the same time, gives more flexibility over the existing solutions. For example, the user may want to query “vehicles” instead of individual sub-types of vehicles such as “cars”, “lorries”, etc. She/he can also query vehicles produced by a specific brand, no matter if it is a car or a lorry. This gives more flexibility than previous query methods, as this gives users more control over the ways of searching for information.

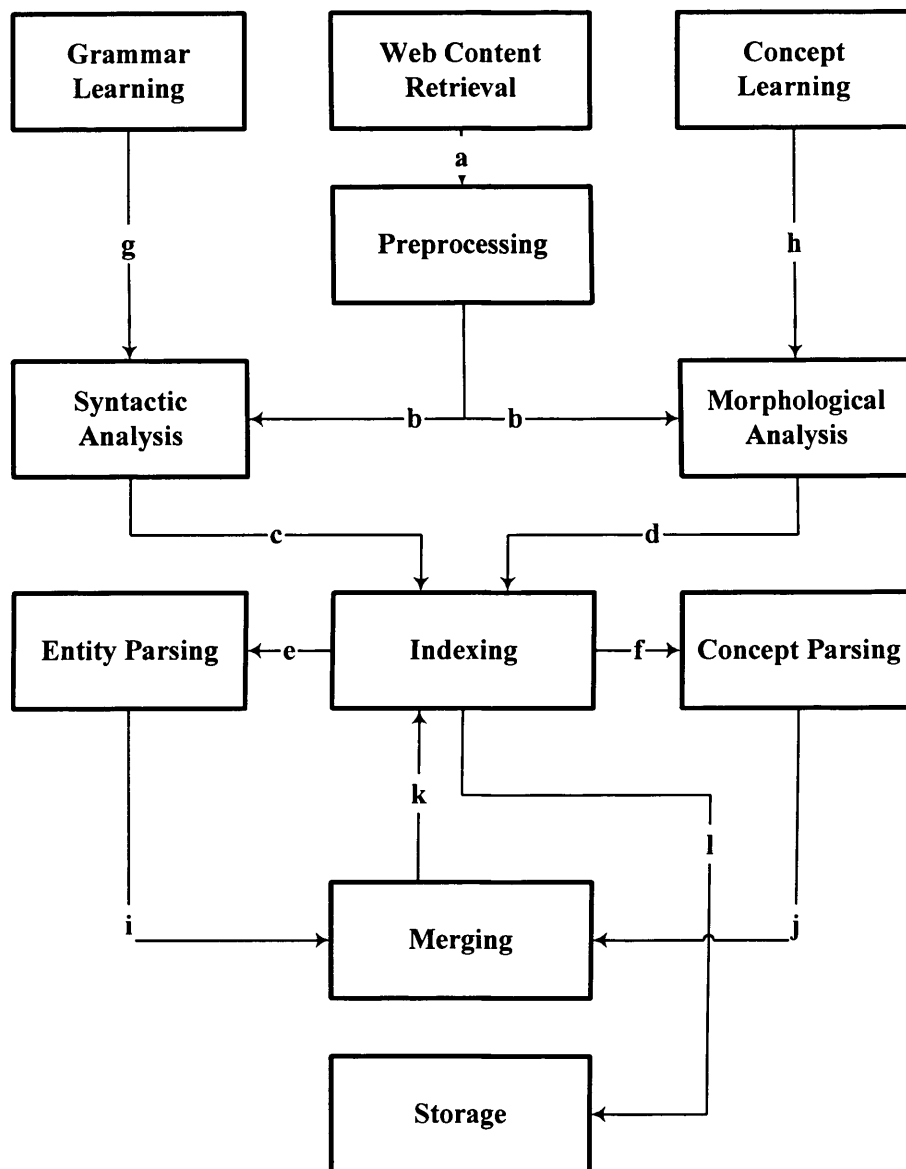
In addition to all requirements defined in section 3.1.2 which have been addressed, there is an extra benefit of using concept indexing for **automatic content categorisation**. The benefit is that different viewpoints, aspects or levels of specification of a domain can be expressed. This benefit is similar to that of using faceted systems [Priss, 2000]. In faceted systems, facets are relational structures consisting of units and relationships that are selected for a certain purpose. Compared with faceted systems, concept indexing has advantages. For example, user can choose abstract concept query terms to form a query, which means that concepts from different levels can be mixed to form a new “facet” which is impossible in faceted systems. Also, in contrast to automatic content categorisation, *no* information is removed from the

original information in the content. This preserves all information necessary to view a document from multiple viewpoints. As a result, knowledge is better reused compared with conventional text clustering and automatic text categorisation methods. The purpose of concept indexing is to preserve the main knowledge of documents' views, so that different facets of a document could be used for different information needs. Using a simple analogy, one can expect that many different questions could be asked against a single sentence, each question focusing on different aspects of the knowledge expressed in this sentence. This issue has not been addressed in previous studies, where documents are normally categorised into static categories, although the static catalogues are generated automatically. In concept indexing, once essential entities and concepts in a text are captured (e.g. agent, place, reason, time, event, status, etc.), the extracted information could be organised in a concept index for more efficient knowledge management.

3.2 A FRAMEWORK FOR CONCEPT INDEXING

In this section an abstract framework for concept indexing is described. The framework is illustrated in Figure 3.3.

There are 11 processes in the framework. The purpose of *web content retrieval* is to collect targeted web pages for further processing. Since vast amount of existing web pages are on the Web, the raw content should be collected in a targeted way. Only web pages of interest will come through to further processing. Irrelevant web pages will not



a - Raw Web Content	e - Indexed Text for Entity Parsing	i - Parsed Entity Information for Indexing
b - Preprocessed Web Content	f - Indexed Text for Concept Parsing	j - Parsed Concept Information for Indexing
c - Syntactic Information	g - Grammar Knowledge	k - Merged Entity and Concept Information for Indexing
d - Morphological Information	h - Concept Knowledge	l - Indexed Merged Entity and Concept Information for Storage

Figure 3.3 Concept Indexing Framework

be stored in the system, thus saving space and processing time and improving the quality of the knowledge extracted by disregarding irrelevant content.

The web pages retrieved from the web are first *preprocessed*, so that all input text material is normalised to one format. Through *syntactic analysis*, lexical information is extracted and syntactic tags [Francis and Kucera, 1979] are added to the free text. The preprocessed text is also used for *morphological analysis*.

Language grammar knowledge and concept knowledge in the framework is provided to conduct *grammar learning* and *concept learning*. These two processes employ supervised machine learning. After training, new knowledge is acquired and stored for further use. Here an assumption is made that knowledge contained in the text used for training and that in the text to be processed are similar. This means that similar patterns are expected from the text to be processed, so that the previously learned knowledge is applicable and could be reused. To give an example, if the training involves material from the engineering field, then the system would unlikely be able to produce accurate results when processing texts from chemistry.

From the syntactic and morphological analysis, text information from both sources is indexed to accelerate the further processing. The processed text with syntactic and morphological information after *indexing* is input to the subsequent *entity parsing* and *concept parsing* to extract entities and concepts. Then, the entity and concept information is *merged* and *indexed* into a structured concept index that is then *stored* for queries.

3.3 ILLUSTRATIVE EXAMPLE

A simplified example illustrating the process of concept indexing is shown in Table 3.2.

The notations used in Table 3.2 (for example, ‘a’ for raw web content, ‘b’ for preprocessed web content, etc.) are the same as those employed in Figure 3.3.

(a) The raw material used in this example is obtained by crawling relevant web sites by targeted web spiders and saved to local hard drive. This material will be used in all subsequent processes. In the example, HTML tags such as “<html>”, “<head>”, “<title>” have special meanings to web browsers such as Netscape navigators and Internet explorers. For instance, “<title>this is the title</title>” indicates that the sentence “this is the title” would appear in the title bar in the windows GUI, while anything between the tag “<body>” and “</body>” will show in the windows, but not in the title bar. These tags are normally interpreted by web browsers but not shown directly to the users.

(b) In the second step, the HTML tags are removed from the raw data (a), and a document reference number is assigned to each document for later use. For example, in Table 3.2, the format of document reference number uses “docnum” followed by 6 digits.

(c) Next, lexical analysis is conducted, and a forward slash “/” and tags indicating lexical ingredients are added after each word. The tags used are the same as those employed in the Brown Corpus [Minnen et al., 2001]. For example, “NP” means proper

Table 3.2 A Simple Illustrative Example of Concept Indexing (a-k)

Content	Data
Raw Web Content (a)	<html><head><title>...</title></head> <body>The Fulton County Grand Jury said Friday an investigation of Atlanta's recent primary election produced no evidence that any irregularities took place.</body></html>
Preprocessed Web Content (b)	The Fulton County Grand Jury said Friday an investigation of Atlanta's recent primary election produced no evidence that any irregularities took place. docnum000001
Lexical Information (c)	the/AT Fulton/NP County/NN Grand/JJ Jury/NN said/VBD Friday/NR an/AT investigation/NN of/IN Atlanta's/NP\$ recent/JJ primary/NN election/NN produced/VBD no/AT evidence/NN that/CS any/DTI irregularities/NNS took/VBD place/NN ./. docnum000001
Morphological Information (d)	the Fulton County Grand Jury say Friday an investigation of one's recent primary election produce no evidence that any irregularities take place . docnum000001
Entity Information (i)	Fulton/NP Country/NN, Jury/NN, Friday/NR, investigation/NN, primary/NN election/NN, evidence/NN, irregularities/NNS, place/NN docnum000001
Concept Information (j)	grand/s586 jury/s272 said/s312 friday/s579 investigation/s26 recent/s19 primary/s362 election/s444 produced/s306 evidence/s306 irregularities/s10 docnum000001
Merged Entity and Concept Information (k)	Fulton/NP Country/NN, Jury/NN, Friday/NR, investigation/NN, primary/NN election/NN, evidence/NN, irregularities/NNS, place/NN grand/s586 jury/s272 said/s312 friday/s579 investigation/s26 recent/s19 primary/s362 election/s444 produced/s306 evidence/s306 irregularities/s10 docnum000001

noun or part of name phrase, while “AT” indicates an article.

(d) Then, inflected forms of verbs in conjugation are primed as in the example. For instance, “said” in (b) “The Fulton County Grand Jury said...” is primed to the original verb form “say”, and so are the others verbs. In addition, possessive and inflected nominal compositions are converted to singular possessive pronouns. For example, “Atlanta's” is primed to “one’s”.

(i) Next, entities are extracted from the indexed content according to the lexical information provided in step (c).

(j) Concept category information is added to the words and phrases in the indexed content. For example, concept category, s586 contained in the concept KB relates to the concept ‘grand’.

(k) In this step, entity information (i) and concept information (j) is combined in a concept index. One word/phrase can show as an entity and a concept at the same time. For example, “Jury” is treated as “singular or mass noun” in the context of entity information, but at the same time, in the context of concept information, “Jury” carries the meaning of concept category “s272”, which is law related. There are some words/phrases which do not appear in (k) as they carry little information. For example, although the word “the” in (c) is treated as an “article”, due to the frequent use of articles in texts, any word with an “article” tag is excluded from (k).

The remaining notions used in Figure 3.3 are implementation-specific. (e) and (f) are

specially structured text facilitating fast entity and concept parsing, respectively. (g) provides grammar knowledge to assist syntactic analysis, and (h) is used to extract concept knowledge from processed Web content. All these specifics will be explained and illustrated in a greater detail in Chapters 5 and 6.

3.4 A SYSTEM FOR CONCEPT INDEXING

3.4.1 System Architecture

The proposed concept indexing system is based on the use of information agents, NLP and indexing techniques. The architecture of the system is shown in Figure 3.3. Web content is retrieved from the web into the system by *web spiders*. The *user queries* and the query results are input into and output from *indexing and query sub-system*. The *web spiders* retrieve web pages from the Web to the internal *indexing and query subsystem*. The *language processor* is used for text preprocessing, parsing syntactic, morphological, entity and concept information contained in the text using *language KB* and *concept KB*. The *indexing and query subsystem* provides three functions to the system, namely, indexing, query and storage functions. It indexes and stores the raw web content, preprocessed, syntactically analysed, conceptually analysed, and merged syntactic and concept information. Syntax and concept information is analysed by the *language processor*. The *concept KB* contains concept knowledge organised in rules and knowledge objects. The *language KB* consists of systematised morphological and syntactic knowledge of English language represented in tagging rules. The *concept*

learning and *syntax learning* modules provide learning ability to the system, so that new concept and grammar knowledge can be obtained by using these two modules.

3.4.2 MAIN PROCESSES

Three processes, **a**, **b** and **c** (see Figure 3.4), are described in this section. The first process (a) relates to retrieving, indexing and storing web content into the system. The second process (b) describes the steps for the system to learn the grammar and concept knowledge needed for processes (a) and (c). The third process (c) concerns the user's interaction with the systems, and involves retrieving relevant knowledge according to the users' queries.

a. Retrieval Process (a). This process begins when the *web spiders* start retrieving web content. This content is stored in the *indexing and query subsystem*'s repository. Before any type of parsing is performed (e.g. syntactic, morphological, entity or concept parsing, as shown in Figure 3.4), text data is preprocessed by the *language processor* to make them normalised for later use. Indexing is carried out to make the processing of the *language processor* faster. Thereafter, the content is analysed by the *language processor* as illustrated in Table 3.2. Entity and concept parsing is performed using information from the *language KB* and *concept KB* when text data from the *web spiders* is preprocessed by the *language processor* and indexed by the *indexing and query subsystem*. After entity and concept information is extracted, it is indexed and added to the concept index for queries. Chapter 4 describes the retrieval process in detail.

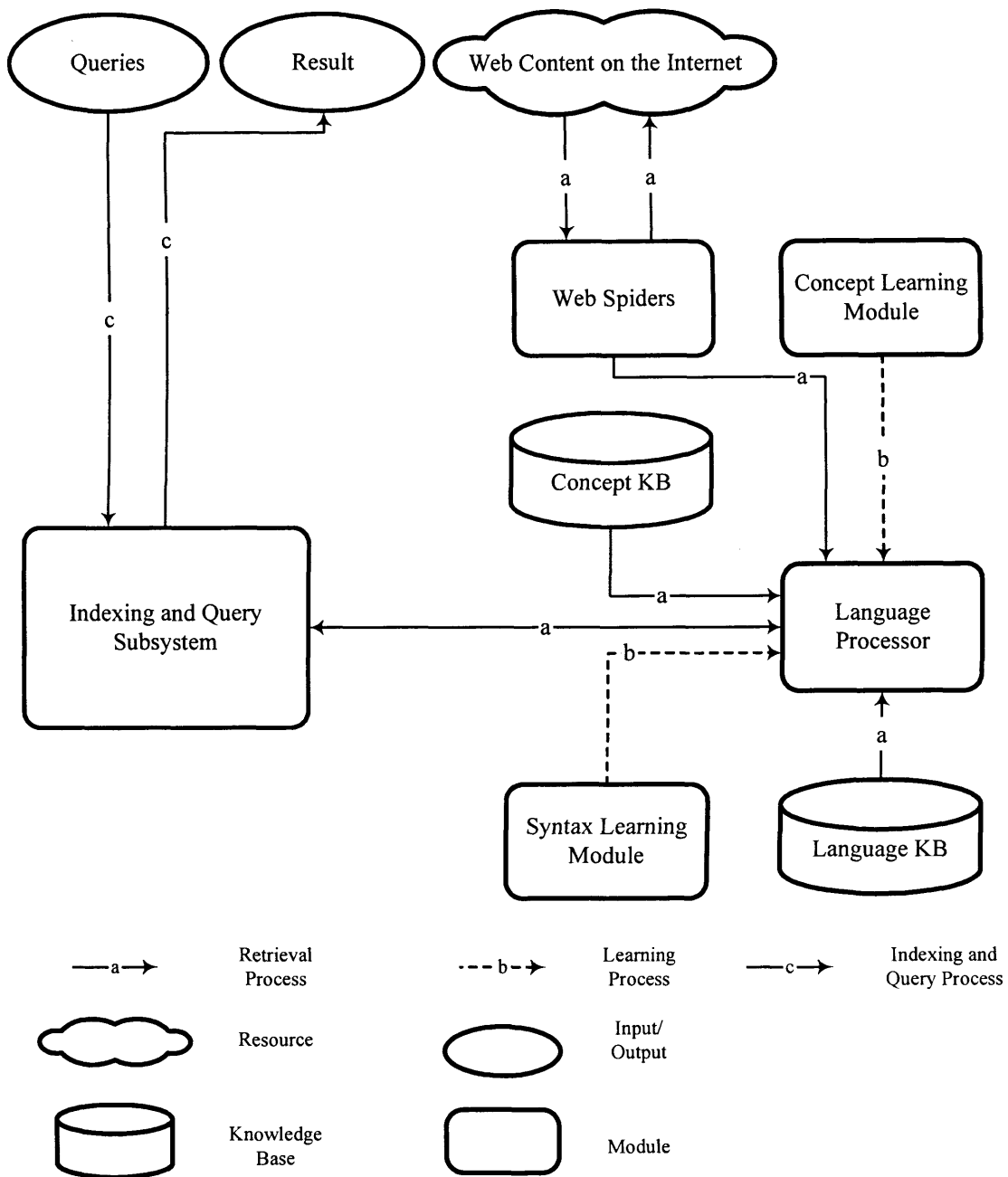


Figure 3.4 General Architecture and Main Processes

b. Learning Process (b). The *language processor* uses knowledge learned in this process to perform the retrieval, indexing and query processes. This process gives the system the ability to automatically acquire new knowledge without building manually the knowledge base. Several machine learning techniques are used in this process which will be explained in more detail in Chapter 5.

c. Indexing and Query Process (c). Users query the system using entities and/or concepts. The indexing is conducted by the *indexing and query subsystem*. The query is passed to this subsystem, which compares it with the entities and concepts contained in the concept index. If the information requested is available, the *indexing and query subsystem* retrieves it from its repositories, and forwards the output to the *concept representation* module for displaying the document set or prompting an error message. Chapter 6 describes the indexing and query process enabling the efficient retrieving of entity and concept information.

3.5 SUMMARY

This chapter has highlighted the problems in the conventional knowledge representation methods and proposed concept indexing, which involves identifying the concepts indicating documents' content, and representing them using a concept index. Entities are extracted from unstructured text-based content using a language knowledge base, and concepts are identified with the help of a concept knowledge base. The entities and concepts isolated are used to build a concept index. The concept indexing

procedure includes the following steps: preprocessing, lexical parsing, morphological parsing, entity parsing, concept parsing and indexing. To illustrate the use of the proposed technique, a concept indexing framework is presented. A system architecture is also developed to illustrate the concept indexing procedures. The proposed approach to knowledge representation will be utilised in Chapters 4-6 with the aim to improve knowledge management.

CHAPTER 4. AUTOMATIC INFORMATION ACQUISITION USING MACHINE LEARNING

In this chapter, the problems associated with finding information on the web are highlighted and a new framework is proposed to address these problems. The framework is based on using intelligent information agents for targeted information retrieval. Next, a machine-learning method is employed to improve the efficiency and accuracy of the retrieval process. Several feature selection options are proposed and evaluated through two case studies from different domains. Finally, tests are conducted to show the applicability of the framework, and the findings are analysed.

4.1 FINDING INFORMATION ON THE WEB

There is a wealth of information on the Web. By 2005, more than eight billion Web pages have been indexed by Google. However, due to the characteristics of the Internet, finding efficiently useful information for knowledge acquisition continues to be a major problem, as the irrelevant Web pages retrieved often waste storage space and computing resources.

Finding useful information on the web is a non-trivial task. This process involves three steps: discovery, retrieval and analysis of the information obtained.

Traditionally, knowledge management uses manual methods for finding, retrieving and verifying the relevance of the information gathered [Wiig, 1995]. Manual analysis is one of the largest bottlenecks in knowledge management [Antoniou and van Harmelen,

2004]. Therefore, new methods for automatic information acquisition are required, which will substantially reduce the cost of the information acquisition process.

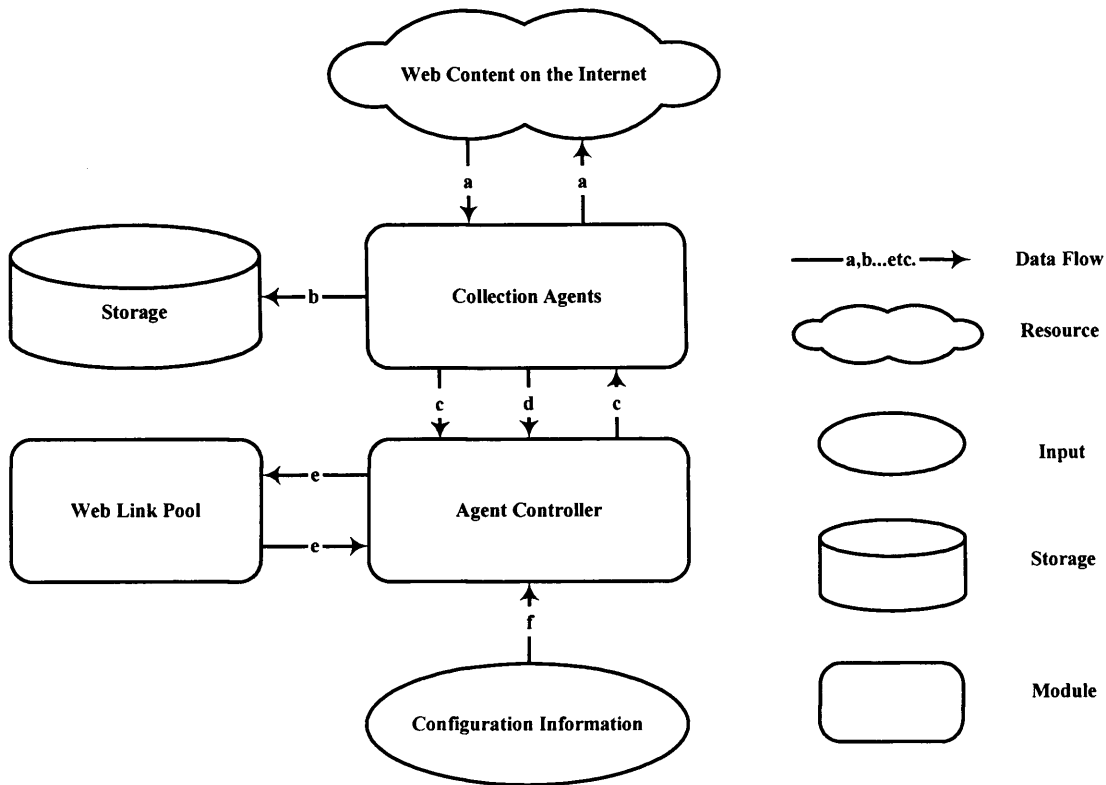
4.2 AN INFORMATION ACQUISITION FRAMEWORK

4.2.1 Structure of the Information Acquisition Framework

The information acquisition framework for collecting relevant information from the Web (Figure 4.1) is composed of four modules: *collection agents*, *agent controller*, *storage*, and *web link pool*.

The task of the *collection agents* is to discover and retrieve relevant information. The *collection agents* communicate with the Web using TCP/IP network protocol, and adhere to the web robot exclusion protocol to ensure that the Web pages are downloaded properly. The *agent controller* manages the coordination between the *collection agents*, ensuring that different agents are assigned different downloading jobs. It also processes the returned web links and stores them in a *web link pool*. In addition, the *agent controller* cancels the tasks finished and assigns new tasks to the agents available. Once a collection of Web pages is returned, the *agent controller* transfers them to the *storage*, which is linked to the module of the *indexing and query subsystem* shown in Figure 3.4.

The data flow shown in Figure 4.1 involves: (a) communication between the *collection agents* and the web, (b) storing the retrieved Web pages into the *storage*; (c) commands sent by the *agent controller* to the *collection agents*, and control feedback information;



a	Network communications
b	Retrieved Web pages into the storage
c	Control commands and control feedback information
d	Web pages returned
e	Web links
f	Configuration information

Figure. 4.1 Information Acquisition Framework

(d) Web pages returned by the *collection agents*; (e) storing newly parsed web links and checking previous stored web links in the *web link pool* by the *agent controller*; and (f) configuration information loaded during the initialisation process.

4.2.2 Information Acquisition Process

The flow charts of the *agent controller* (Figure 4.2) and the *collection agents* (Figure 4.3) illustrate the information acquisition process.

The process begins when the system initialises itself by loading the configuration information from the configuration file. Some web links need to be provided at this stage as ‘seed’ web addresses. After initialisation, the *agent controller* stores these web addresses into the *web link pool*. The *agent controller* then generates *collection agents*, and assigns them a set of Web links which they have to explore. When an agent finishes its task, the *agent controller* stores the Web pages and links returned by the agent into the *Web link pool*, and assigns the agent a new task (a new Web link to explore from the *web link pool*). The process finishes when certain termination conditions (such as number of Web pages retrieved or retrieval time) are reached.

In this framework, the relevance of the documents retrieved depends on the keywords supplied in the configuration file, which is used during the initialisation process. The agents determine the relevance of the retrieved Web pages based on whether these keywords appear in them. If more than one keyword is set in the configuration file,

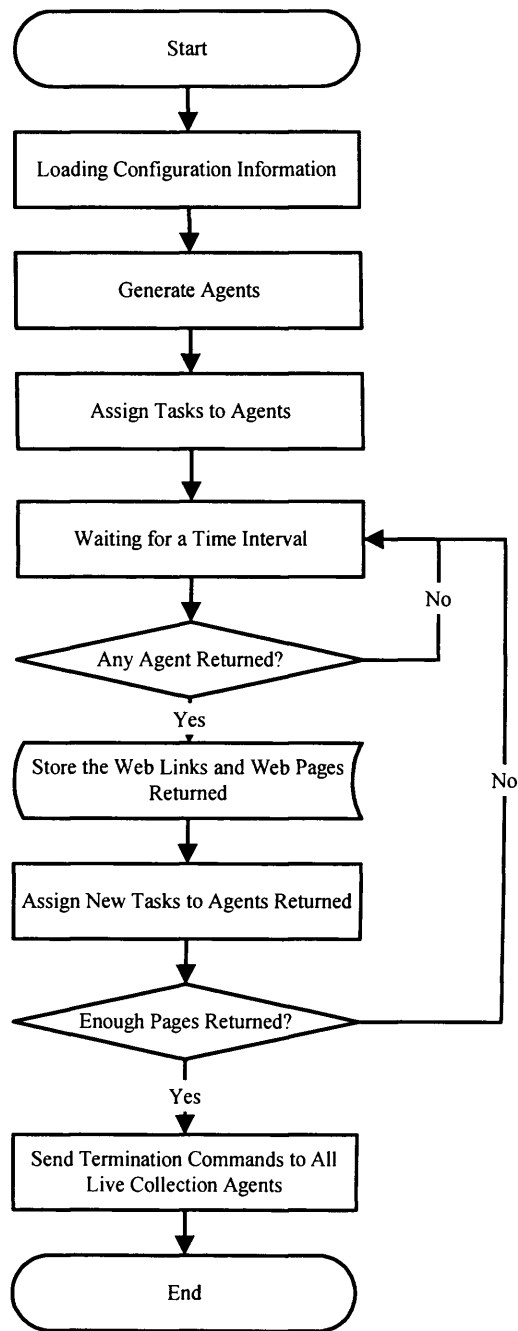


Figure 4.2 Flow Chart of Agent Controller

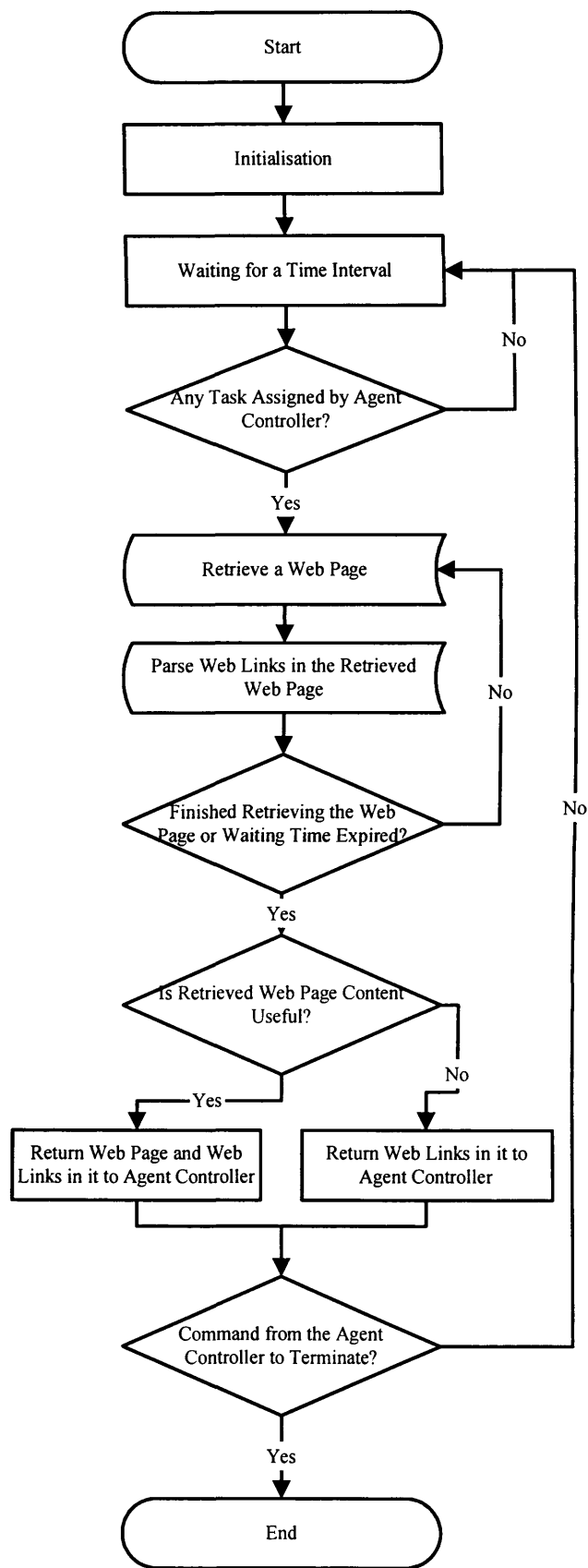


Figure 4.3 Flow Chart of a Collection Agent

then the user has the choice whether the agent should return pages containing either one of the keywords or only those that have every keyword. If the retrieved Web page does not contain the keywords, then only the web links contained in it will be returned and stored in the *web link pool* for later use.

4.3 AN IMPROVED INFORMATION ACQUISITION FRAMEWORK

4.3.1 Using Support Vector Machine (SVM) for Information Retrieval

The information acquisition process depicted in section 4.2 is dependent on the keywords provided. The problem associated with this approach is that both relevant and irrelevant Web pages may contain the same keywords. Therefore, some irrelevant Web pages will still pass through the agents, and be stored into the system. Obviously, this would reduce the system performance in terms of precision.

For example, suppose a user would like to build a knowledge base about ‘fishing’, and he/she is interested in fishing as a hobby. However, not all Web pages which contain the keyword “fishing” would be relevant. Some of them may refer to ‘fishing industry’ and ‘fishing product sales’. If such Web pages are stored into the system, the quality of the fishing knowledge base would be substantially reduced.

It is proposed to use a data classification method, Support Vector Machine (SVM), to improve the precision of the retrieval process and enable the building of knowledge

bases automatically from the Web.

To understand the main approach of SVM, the concept of *hyperplane* needs to be introduced. In n-dimensional Euclidean Space R^n , the set S is called a *hyperplane*, when S satisfies the condition in (4.1) [Weisstein, 2005].

$$\text{set } S = \{ X = [x_1, x_2, \dots, x_n]^T \mid a_1x_1 + a_2x_2 + \dots + a_nx_n = 0 \} \quad (4.1)$$

where

a_1, a_2, \dots, a_n are scalars which is not all equal to 0, X is a vector.

The purpose of SVM is to find a *hyperplane* that leaves the largest possible fraction of vectors of a same class on the same side, and maximises the distance of both classes from the hyperplane. When data is not separable in the original input vector space using a hyperplane, it needs to be transformed from the input vector space into a higher dimensional space, which is called a *feature space*. The transformation function is called a *kernel function*. If the kernel function is a linear function, then the SVM uses a *linear kernel*, if a non-linear function such as sigmoid or radial basis function is used, the SVM uses a *non-linear kernel* [Schölkopf and Smola, 2002].

As stated by [Joachims, 1998] and [Dumais, 1998], SVMs with linear kernels outperform many traditional text classifiers. In addition, it has been reported that nonlinear SVMs (e.g. those using high dimension polynomial and radial basis kernels) are much more complex and provide insignificant improvement in classification accuracy compared to those using linear kernels. Therefore, SVM with a linear kernel is

proposed as the machine learning technique employed to improve precision.

When the hyperplane is determined by training, and the data instances that need to be classified are expressed as vectors, the classification can be performed by calculating the parameter C that is determined through (4.2). If C is a non-negative value, then the instance is classified as an instance of a positive class, otherwise it is considered as belonging to a negative class.

$$C = x_i \cdot w + b \quad (4.2)$$

where

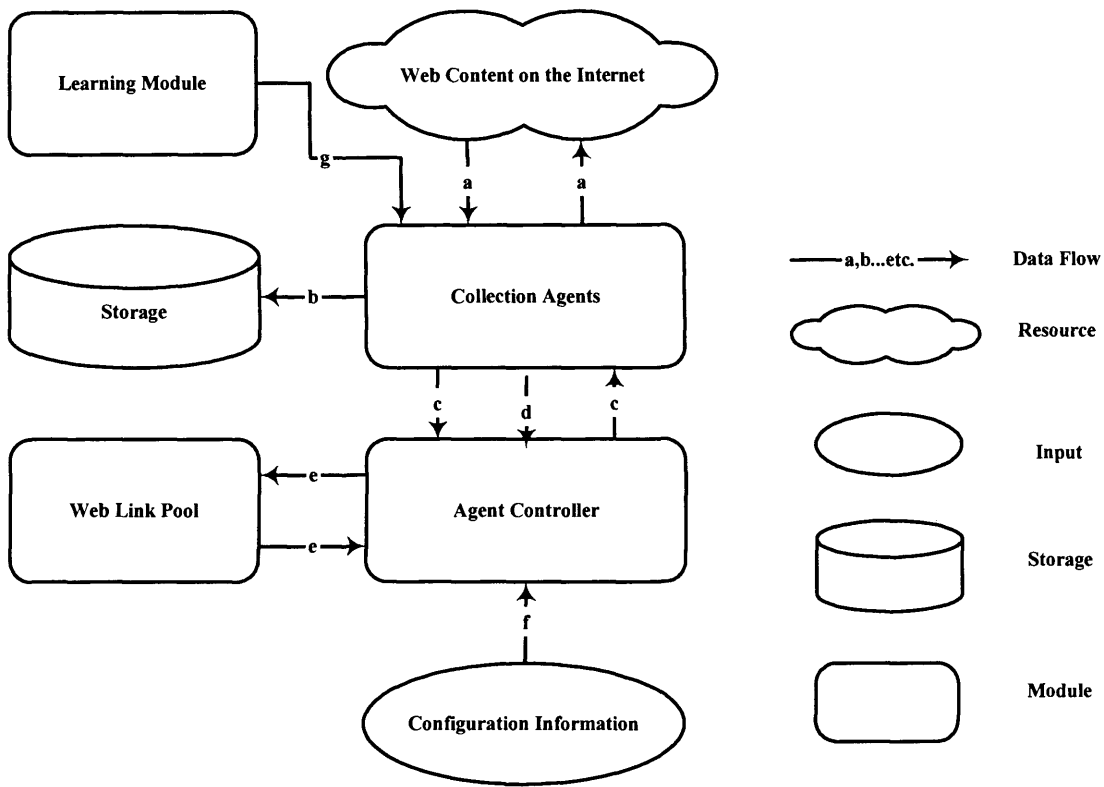
C is a real number;

x_i is the instance to be classified, expressed as a vector;

w and b are parameters describing the hyperplane that are determined during the training.

4.3.2 Machine Learning in the Improved Information Acquisition Framework

The improved information acquisition framework, which uses SVM is shown in Figure 4.4. Note the *learning module* added to the system which provides the system with learning ability. During its training, the module is supplied with relevant and irrelevant HTML pages which serve as positive and negative examples for training. Each HTML pages is represented as a vector, with elements called *features*. The SVM obtains the



a	Network communications
b	Retrieved Web pages into the storage
c	Control commands and control feedback information
d	Web pages returned
e	Web links
f	Configuration information
g	Knowledge obtained to help filter Web pages returned

Figure 4.4 Improved Information Acquisition Framework

statistics from those vectors in the training examples, and then transfers this information to the *collection agents* module, so that the agents can use it to determine the relevance of the Web pages they retrieve. Compared to the basic framework which uses a keyword matching method, this framework is expected to be much more effective in situations when both the relevant and irrelevant Web pages contain the same keywords, in which case, the basic framework will wrongly consider all these Web pages relevant.

The machine learning process used in this chapter includes:

- 1. Obtaining examples.** First, training examples with known output results are provided as prior knowledge so that the knowledge obtained in the training process can be later applied to unseen instances.
- 2. Selection of appropriate features.** In SVM, the characteristics of the training examples are represented as features. Therefore, the question is what features best represent discriminative knowledge so that the highest classification rate possible is achieved.
- 3. Generating data sets for training and testing purposes.** In this step, the examples selected in step 1 are converted according to the features selected in step 2 into data sets. This step involves preprocessing procedures needed to convert the training examples into a format suitable for a specific implementation of the machine learning algorithm. The values included in the training sets depend on the features selected in step 2; different features would generate different training sets, even if the

same training examples are used.

4. Training. The training data sets created in step 3 are used as input to the machine learning algorithm which determines discriminative patterns during the training process.

5. Applying knowledge to unseen instances. This step uses the patterns obtained in step 4 to classify unseen instances.

Obviously, the feature selection step is critical in achieving high classification accuracy.

The next section discusses how to select appropriate features to represent Web pages, so that they can be accurately classified.

4.3.3 Feature Selection for the Proposed Machine Learning Algorithm

In this section, three options for feature selection are discussed. These are using (i) HTML element attributes, (ii) a small or bigger general dictionary, and/or (iii) words lemmatisation.

1. Using HTML element attributes as features.

In general, an HTML document is a sequence of characters organised physically into a set of entities, and logically as a hierarchy of elements. In an HTML document, an element refers to a component of the hierarchical structure defined by a document type definition (DTD); it is identified in a document instance by descriptive markup, usually

a start-tag and end-tag. In a start-tag, white space and attributes are allowed between the element name and the closing delimiter. An attribute specification typically consists of an attribute name, an equal sign, and a value, though some attribute specifications may be just a name token.

A simple HTML file (Figure 4.5) is used to illustrate this. In line 5 of this fragment, the HTML element is “<meta http-equiv=“keywords” content=“Fitting Jobs”>”, and its name is “meta”. The element contains two HTML attributes, named “http-equiv” and “content”, and their attribute values are “keywords” and “Fitting Jobs”. The advantage of employing element attributes is the expected improved accuracy as very often the HTML element attributes contain meaningful words. However, some HTML attributes are too frequently used, and this may lead to wasting storage space and computing resources, and reducing the speed of the retrieval.

2. Using words from a purpose-built small dictionary or a general dictionary with a relatively large vocabulary.

The user could choose to compose a dictionary that only contains specialist terms such as “fish”, “tide”, “fisher”, “salmon” etc. As the dictionary is normally small, dedicated, and containing only relevant words, the processing speed would be very high. However, the user may add less important words and at the same time miss some essential ones, which will inevitably affect the classification accuracy.

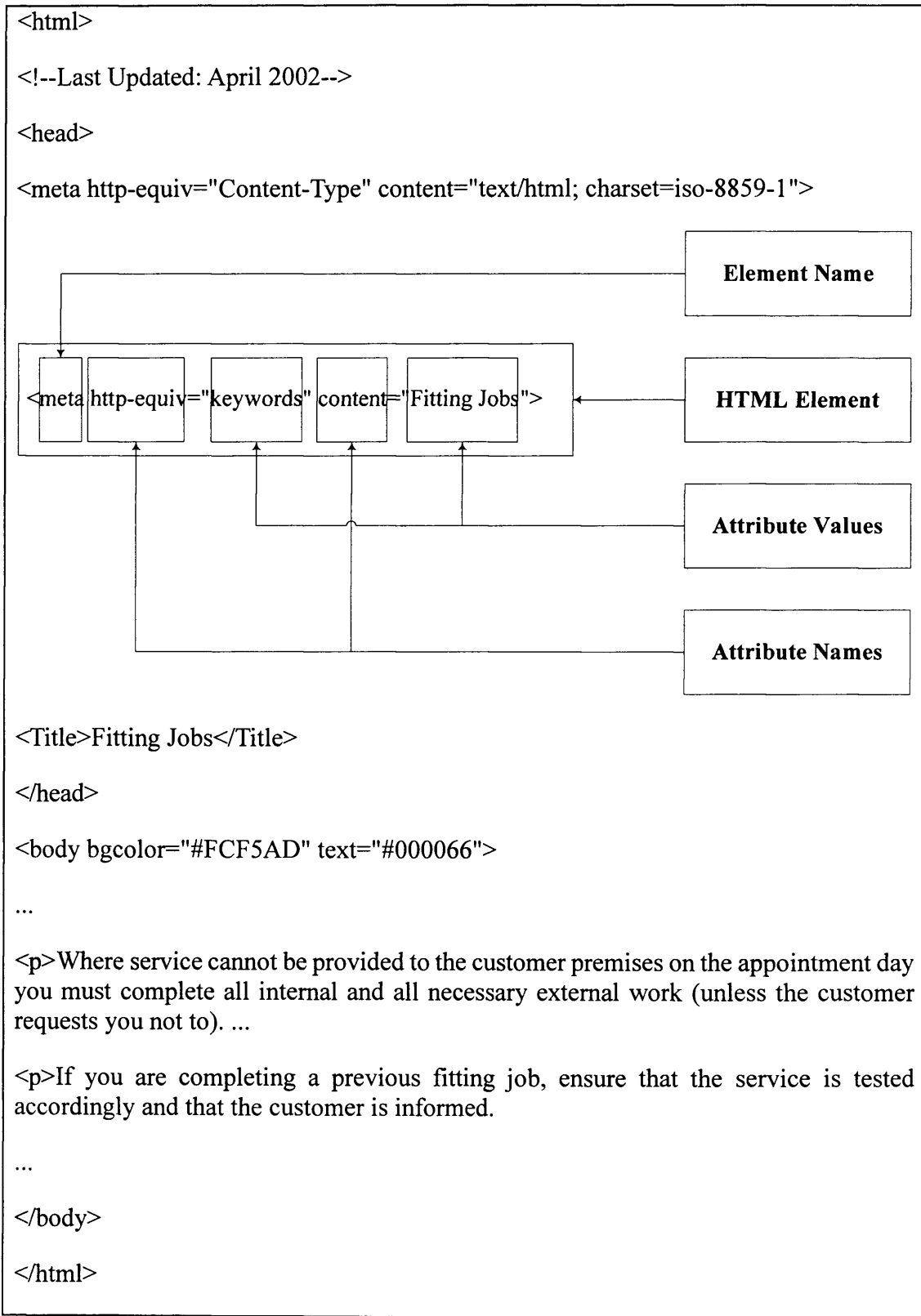


Figure 4.5 An Example HTML File Annotated

3. Using words lemmatisation¹

The words in the HTML pages are often different from their lemmas², especially when nouns and verbs are concerned. For instance, the lemma of “running” and “gave”, respectively, are ‘run’ and “give”. The use of lemmatisation allows otherwise ignored words to be utilised in building the feature space. However, the use of lemmatisation may lead to including incorrect features in the feature space. For example, after lemmatisation, “Warner Brothers” changes to “Warner Brother” which loses its original meaning. As a result, that HTML page might be wrongly classified.

4.3.4 Case Studies

General Approach

Two case studies are conducted to compare the three options considered in terms of their classification accuracy which is measured using *precision* and *recall*. In addition, the processing time is also recorded, as it is an important measure of performance when classification algorithms are compared. The first case study is on a popular subject (fishing), while the second one is chosen from the domain of engineering. The procedure explained below follows the machine learning process outlined in section 4.3.2. The procedure is illustrated through examples included in Appendix A.

¹ Lemmatisation refers to the process of grouping the inflected forms of a word together under a base form, or of recovering the base form from an inflected form, e.g. grouping the inflected forms ‘run’, ‘runs’, ‘running’, ‘ran’ under the base form ‘run’ [Mitkov, 2003].

² Lemma refers to the canonical form of a word, usually the base form, taken as being representative of all the various forms of a morphological paradigm [Mitkov, 2003].

First, a number of Web pages are retrieved by the framework described in section 4.2. These pages are marked by a human expert as positive and negative examples, according to different criteria used in each case study. Example pages related to case study 1 are shown in Appendix A.1.

Then, the retrieved web pages are pre-processed according to the features selected in each test. For example, when HTML element attributes are used as features, the Web page marked as a positive example in A.1 of Appendix A is retrieved and saved in the format shown in A.2. If the option is not to use HTML element attributes as features, then the saved Web page in A.2 undergoes additional processing where the HTML tags are stripped from the text, as shown in A.3.

Next, the Web pages in their original HTML form (as in A.2) or after pre-processing (see A.3), depending on the option chosen, are converted into a training data set. For example, the text in A.3 is converted to a vector of features as shown in (4.3). The format of a feature employed in this thesis is adopted from the implementation of SVM^{light} [Joachims, 2002].

+1 2:0.0080 7:0.0080 36:0.0080 59:0.04 62:0.048 111:0.0080 146:0.0080 #	(4.3)
---	-------

Each feature (4.4) is characterised by its “feature ID” and “feature value”. “Feature ID” is a unique integer number which identifies each feature.

(Feature ID):(Feature Value)	(4.4)
------------------------------	-------

“Feature Value” (4.5) is a real number, which is defined as the ratio of the word

frequency and the number of words in a processed Web page. A colon “:” is used to separate the feature number and the feature value.

Feature Value = $(m)/(n)$	(4.5)
---------------------------	-------

where,

m is the word frequency (i.e. the number of times a particular word appears in a given text),

n is the total number of unique words in a processed Web page.

In the vector shown in (4.3), “+1”³ indicates a positive training example which is considered relevant. “2:0.0080” suggests that the 2nd feature has a feature value of “0.0080”, while “62:0.048” indicates that the 62nd feature has a feature value of “0.048”. A feature with zero value does not show in the training sets to save storage space, but it is still counted in the training process as having a zero value. For example, features 3-6 between 2 and 7 have zero feature values and are represented as “4:0.0”, “5:0.0” and so on. In all files produced by SVM^{light}, data or characters after “#” are ignored as comments.

The following example illustrates how feature values are calculated. The word “alive” appears in the small dictionary (Appendix A.4) used in the first case study at 2nd place, which determines its feature ID as ‘2’. It appears once in the processed Web page (Appendix A.3), which includes unique 125 words. Therefore, the feature value for

³ In the case of a negative example, “-1” is used instead of “+1” but the format of the rest remains the same.

feature number 2 is $1/125 = 0.008$. In the training set, records like (4.6) below are added for each feature used.

2:0.0080	(4.6)
----------	-------

However, if a word appears in the processed Web page, but not in the dictionary, then it is not considered as a feature. For example, the word “museum” in the retrieved Web page (Appendix A.3) is not used as a feature, as it is not contained in the small dictionary (Appendix A.4).

After the processing, the training data is input to the SVM^{light} [Joachims, 2002] to generate a data file as illustrated in Appendix A.5. The file generated contains information, which is further used to calculate the value of C as in (4.2) for each instance in the testing data set where the value is used to classify whether the testing instance is relevant or not. If the float value calculated is a non-negative one, then the corresponding testing instance is classified as relevant, and otherwise it is considered irrelevant. Finally, the results generated from the testing data set are examined to measure the classification performance in terms of precision, recall and processing speed.

All tests were conducted on a Pentium III 700MHz 384MB memory computer with Java Virtual Machine version 1.4.2 with default settings.

Case Study 1: Fishing

A popular subject, “fishing”, was chosen as one of training scenarios examined with the

help of a domain expert. 100 Web pages retrieved by the system described in section 4.2, were used as materials for training and testing. These pages, containing at least once the keyword “fishing”, were analysed by the domain expert using the following criteria: (1) The main content of the Web page must be relevant to “fishing” as a hobby; (2) It must not describe product sales for fishing; (3) It must not be concerned with the fishing industry; (4) It must predominantly contain text, not images; (5) It must not be generated by a search engine. This ensures that the content is retrieved from actual web sites, and does not contain mainly pointers to web sites generated by a search engine.

The 100 Web pages were manually classified as positive and negative examples, and split into a training set and a testing set. The training set consists of 62 examples, which are numbered from 1 to 62, and the testing set has 38 examples, from number 63 to 100. The training set includes randomly chosen 32 positive and 30 negative examples while the testing set contains the remaining 14 positive and 24 negative examples.

183 keywords for fishing were selected by a domain expert to build a domain specific dictionary, e.g. a *small dictionary* (SD) (A.4). The words selected in SD are chosen from the 46 positive examples included in the 100 Web pages retrieved. In addition, a *big dictionary* (BD) was built which includes 9947 words from a TOEFL⁴ vocabulary dictionary as a general keyword dictionary.

Five tests were conducted to investigate the effect of the options discussed in section 4.3.3, as follows:

⁴ TOEFL stands for “Test of English as a Foreign Language”.

- TEST1. Use of HTML element attributes and SD
- TEST2. Use of HTML element attributes and BD
- TEST3. Use of SD without using HTML element attributes
- TEST4. Use of BD without using HTML element attributes
- TEST5. Use of BD and lemmatisation without using HTML element attributes

For TESTS 3-5, the same HTML pages used in TESTS 1 and 2 had their HTML attributes removed from both the training and testing sets. TEST 4 required morphological analysis to be first conducted after which the words were transformed to their lemma forms.

The experimental results are shown in Table A.2 of Appendix A.6. The table contains C values calculated by SVM^{light} for each Web page included in the testing set (from number 63 to 100). These values are used to differentiate between relevant and irrelevant pages. A non-negative value (such as 0.3395136 or 0.25341148) indicates a relevant page while a negative value (e.g. -0.66350608 or -0.75570562) suggests that the corresponding Web page might be irrelevant. These values do not represent the degree of relevance or irrelevance; they are just used to classify the Web pages in two categories. Table A.3 (Appendix A.6) contains the same results as Table A.2, with the non-negative values replaced by P (positive), and the negative values substituted by N (negative). In addition, table A.3 indicates the opinion of an expert about each of these pages, which is further used to measure whether a particular page has been classified

correctly by the system. As stated previously, precision and recall are used to measure the classification accuracy of the system for the purpose of information retrieval. For example, the precision in TEST 1 (Table A.3) is 50%. It is calculated as the ratio of the number of documents, which are correctly classified as relevant (2 instances), and the number of all documents retrieved (4 instances). The recall in TEST 1 (14.29%) is calculated as the ratio of the number of correctly identified documents (2 instances) and the number of all relevant documents as classified by the domain expert (14 instances).

The results are summarised in Table 4.1. Among all five tests conducted, the lowest precision (50%) and recall (14.29%) is in TEST 1 when HTML element attributes and a SD are used. In TEST 5, the use of a BD and lemmatisation (without using HTML element attributes) provides the highest precision (77.78%), while the use of HTML element attributes and a BD in TEST 2 reveals the highest recall (78.57%).

The comparison of the results in TEST 1 and TEST 2 shows higher precision (61.11%) and higher recall (78.57%) when a BD is employed. A similar comparison between TEST 3 and TEST 4 results reveals some decrease in precision (from 71.43% to 66.67%) and increase in recall (from 35.71% to 42.86%).

Fastest processing speed (2 seconds) was recorded in TEST 1 when using HTML element attributes and a SD. The most time consuming test was TEST 5 (205 seconds) involving a BD and lemmatisation, and no HTML element attributes. For obvious reasons, the processing is faster when using SD dictionary instead of BD, with all other conditions remaining the same.

Table 4.1 Summary Results for Case Study 1

Test	Test Conditions	Precision (%)	Recall (%)	Processing Time (seconds)
TEST 1	Use of HTML element attributes and SD	50	14.29	2
TEST 2	Use of HTML element attributes and BD	61.11	78.57	13
TEST 3	Use of SD without using HTML element attributes	71.43	35.71	191
TEST 4	Use of BD without using HTML element attributes	66.67	42.86	196
TEST 5	Use of BD and lemmatisation without using HTML element attributes	77.78	50	205

Case study 1 shows that the use of a specially designed SD or general BD influences precision and recall. Using a general BD can improve recall while the use of a SD could make the processing faster although it would take time for a domain expert to build a dictionary. In addition, there is always a possibility of omitting some important words from the dictionary which means that not all significant features would be considered during the training. Compared to SD, the use of a BD allows more features to be included in the training set, which reduces the demand on the quality of the custom made dictionary. This analysis indicates that the use of a general BD may be a good option in terms of saving development time and producing good results. However, more tests are needed to ensure that this conclusion is applicable to other domains and valid for larger data sets.

With regard to lemmatisation, the results of TEST 5 in comparison with TEST 4 show that both precision and recall are improved by 11.11% and 7.14% respectively. The positive effect on the precision and recall is due to the use of additional lemmatised words, which have been originally excluded, “visible” to the SVM.

In general, the results of TEST 2, 4 and 5 are more promising than those of TEST 6 and 7. This is attributed to the use of a BD which will be further investigated throughout the second case study.

Case Study 2: Site Engineering

In this case study, Web pages are retrieved from a LAN web directory, which contains

information for field engineering, electronic maintenance, general processes, health and safety, vehicle and engineering product support information. These Web pages are maintained by domain experts. Two categories are chosen for case study 2: *processes* and *tasks*. *Processes* here refer to procedures used in field service on site such as pole testing, repair process and fitting jobs. There are 65 Web pages related to this category. *Tasks* refer to information about products and services, such as network routing service and wiring service. The collection contains 44 pages with information about *tasks*.

For the purpose of this experiment, all pages related to the first category (*processes*) are considered as positive examples, and all pages from the second category (*tasks*) are classified as negative examples. All these 109 Web pages are split into two sets: 89 Web pages are used for training and the remaining 20 pages for testing.

Two more tests were conducted to further investigate the effect of the options discussed in section 4.3.3. These are:

- TEST 6. Use of BD without using HTML element attributes
- TEST 7. Use of BD without using HTML element attributes and lemmatisation

The experimental results shown in Tables A.4 and A.5 of Appendix A are summarised in Table 4.2. As shown, TEST 7 which investigates the effect of lemmatisation, provides better precision and recall but is more time consuming. This means that when lemmatisation is used, more useful features are introduced into the system. Although this may lead to including some irrelevant features, it would not necessarily produce

Table 4.2 Summary Results for Case Study 2

Test	Test Conditions	Precision (%)	Recall (%)	Processing Time (seconds)
TEST 6	HTML element attributes and BD	69.23	90	185
TEST 7	HTML element attributes and BD and lemmatisation	71.43	100	198

lower precision. The 100% recall rate in TEST 7 means that all “positive” Web pages have been successfully identified. This result suggests that the use of lemmatisation may lead to better precision and recall.

Discussion

Based on the two case studies conducted, the following recommendation are made for the use of the SVM algorithm for information retrieval.

1. If the processing speed is a concern, then a carefully built customised dictionary should be used.
2. If time and cost to build a special domain-specific dictionary is a concern, then a general dictionary with large vocabulary could be a good choice.
3. Using a general dictionary and lemmatising the words in Web pages could be a good option to achieve good precision and recall.

4.4 SYSTEM DESIGN AND FUNCTION DESIGN

The system is implemented in Java with the use of two external software components: SVM^{light} [Joachims, 2002] and a lemmatiser [Minnen et al., 2001]. The UML class diagram (Figure 4.6) shows the relationships between the classes used in the design of the information retrieval system whose framework was presented in section 4.3.2. These classes are designed to implement the functions of the modules shown in Figure 4.4.

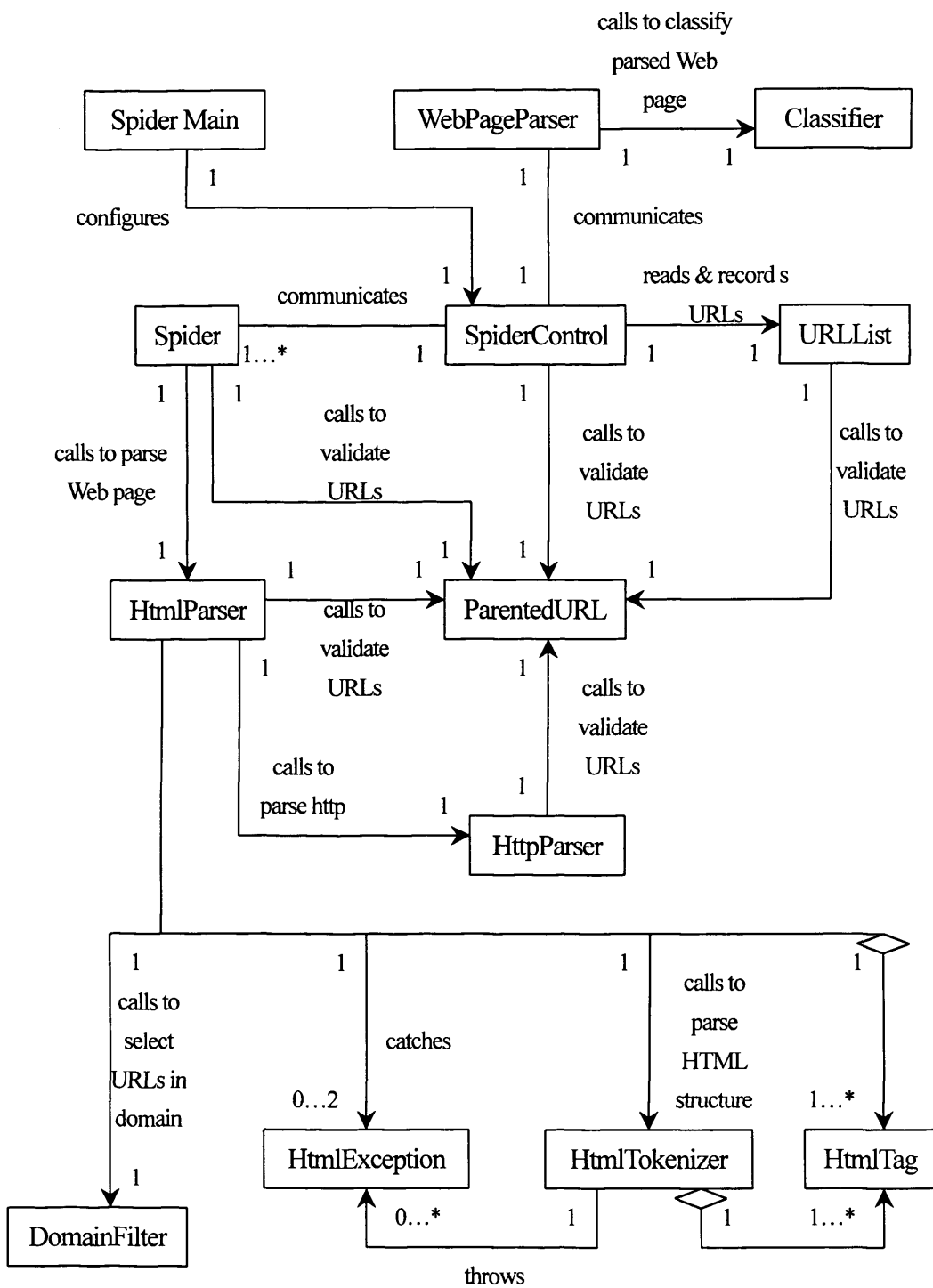
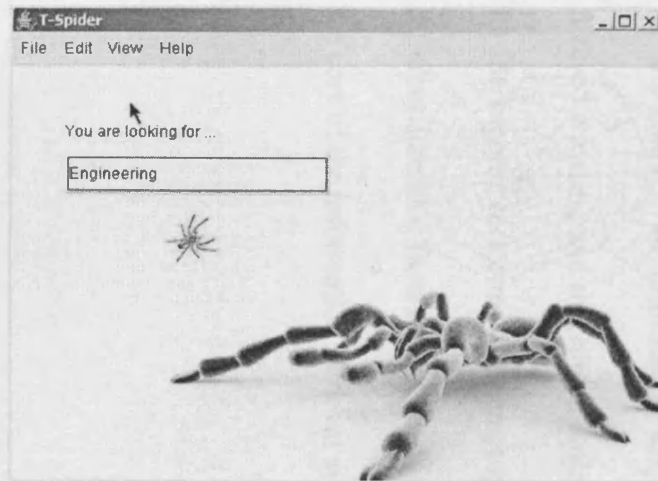


Figure 4.6 UML Class Diagram of the Improved Information Acquisition System

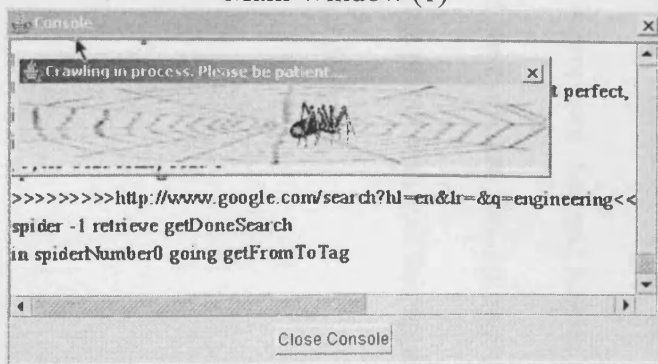
SpiderMain (Figure 4.6) serves as the main programme, which loads the configuration function and controls the main workflow of the system. *SpiderControl* coordinates the retrieving tasks of the *spiders*, ensuring that they work efficiently. *URLList* stores the web links parsed from the Web pages processed so that *SpiderControl* can assign unretrieved Web links to the *spiders* to explore. *WebPageParser* and *Classifier* are used to determine whether a particular Web page is relevant or not. In addition, there are classes, which provide useful functions to other classes. These are *HtmlParser*, *ParentedURL*, *HttpParser*, *DomainFilter*, *HtmlException*, *HtmlTokenizer* and *HtmlTag*. Their purpose is depicted through the annotated links in the UML diagram. A small part of the source code developed is included in Appendix B.

The system function design is illustrated through the screenshots of the graphical user interface shown in Figure 4.7. The user starts the information retrieval process after providing the system with keywords (1), a storage location (2) and a starting URL (2). If a retrieved Web page contains all keywords supplied through (1), it is stored in the location given in (2). The starting URL (2) is used to retrieve the first Web page, after which more URLs parsed from this and other Web pages are added to the *URLList* for further use.

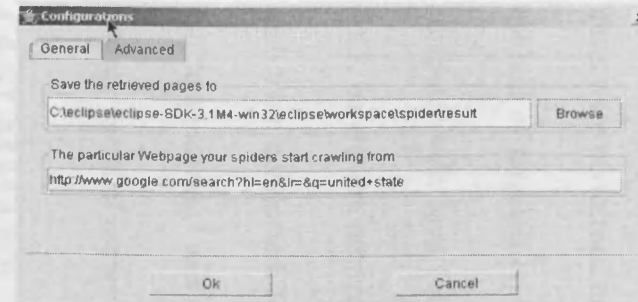
Furthermore, additional options are provided to “fine tune” the system using the advanced configuration window (3). For example, the user can specify the number of spiders working concurrently, the number of Web pages to be retrieved and the length of time a spider should wait for to retrieve any Web page. The configuration settings



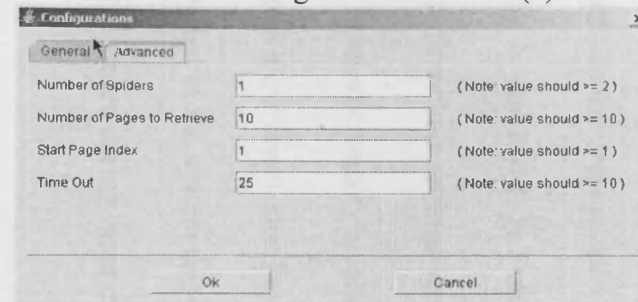
Main Window (1)



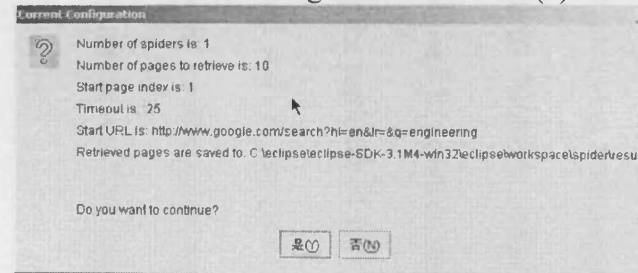
Runtime Window (5)



General Configuration Window (2)



Advanced Configuration Window (3)



Configuration Confirmation Dialogue Box (4)

Figure 4.7 Screenshots of the Graphical User Interface

provided to the system through windows (2) and (3) are confirmed in (4). In addition, the system run time status is shown using (5).

Tests were performed to measure the average speed of retrieving Web pages when this method of targeted crawling is employed. A Pentium III 700MHz 384MB memory computer with Java 1.4.2 with default settings was utilised to conduct the tests on a 300K bandwidth network. Five experiments involving the retrieval of 100 Web pages each were conducted. In each experiment, one keyword such as “seagull”, “fishing”, and “engineering” was supplied to the system through the configuration window. The average speed of retrieving recorded during these experiments was 2.64 page/second.

4.5 SUMMARY

In this chapter, an information acquisition framework based on using intelligent focused crawling and machine learning is proposed. Several feature selection options for the support vector machine employed are evaluated through two case studies from different domains. Tests conducted show the practicability of this approach, which is in the core of the information retrieval system designed.

CHAPTER 5. ONTOLOGY DICTIONARY AND CORPUS FOR ONTOLOGY TAGGING

In this chapter, the problem of ontology tagging (OT) is discussed, its purpose and importance are explained. The resources enabling ontology tagging are identified; these are an ontology dictionary and an ontologically tagged corpus. First, the procedure to build an ontology dictionary is described. Next, the steps to build an ontologically tagged corpus are identified. To construct the corpus, first an electrical dictionary is created, and then a heuristic approach, which generates a mapping between the entries in this dictionary and the ontology dictionary, is developed and tested. Finally, a well-known and widely used corpus is converted into a tagged ontology corpus by utilising the mapping algorithm developed. The ontology dictionary and the ontologically tagged corpus created will be further exploited for ontology tagging of texts as described in Chapter 6.

5.1 ONTOLOGY TAGGING

In this research, a ‘*tag*’ refers to a label used for categorisation and discrimination, which expresses the meaning of a piece of text. For example, the HTML tags contain information used by Web browsers to render the content of the Web pages displayed, e.g. any text between “” and “” is displayed in bold. “*Tagging*” refers to the process of assigning tags to text so that it could be processed and “understood” by

machines. For example, part of speech¹ (POS) tagging refers to the automatic assignment of tags to grammatical classes of words, such as nouns and verbs. POS tagging is used for various language processing tasks, linguistic studies, information technology applications, and speech processing [Voutilainen, 2003].

Recent research indicates that the use of ontology could aid many applications related to natural language processing (NLP), such as information retrieval, information extraction, text summarisation, semantic similarity and word-sense disambiguation. [Vossen, 2003]. In the context of this work, *ontology tags* are defined as a set of systematically defined labels, which express ontology information; *ontology tagging* (OT) is referred to the process of assigning ontology tags to words and phrases within a text, for the purpose of exploiting the ontology information in various applications.

The benefits of employing OT are twofold. First, they allow text to be searched, browsed and analysed at different abstraction levels. To give an example, a text about Ferrari sports cars and F1 cars can be viewed at different abstraction levels, starting from “sports cars”, “cars”, “vehicles”, to more abstracted “transportation means”. This is similar to browsing a digital map, where one can ‘zoom in’ for details or ‘zoom out’ for a general view.

The second benefit is associated with the automatic creation of RDF based data models

¹ Part of speech in linguistics refers to any of the basic grammatical classes of words, such as noun, verb, adjective, and preposition [Mitkov, 2003].

for the Semantic Web by using existing text-based document collections. As indicated by [Benjamins et al., 2002], creating a vast amount of new content and the potential risk of excluding existing Web content from the Semantic Web are serious concerns that challenge the Semantic Web. However, the use of automatic ontology tagging may help to address these challenges, as the existing Web content can be automatically made appropriate to be used on the Semantic Web by utilising OT. Thus, by reusing existing Web content, the cost of creating new Semantic Web content could be substantially reduced.

Despite its benefits, ontology tagging is currently considered impractical due to the high cost associated with manual ontology tagging. Therefore, an automatic ontology tagging method is needed which automatically acquires ontology knowledge from training material by using machine learning (ML). Machine-readable dictionaries (MRD)² and corpora³ [Matsumoto, 2003] are typical training materials for NLP purposes. Currently, there are no such training materials available for ontology tagging. Therefore, there is a need for developing:

- (i) An *ontology dictionary* with standardised ontology definitions of large quantity of words and phrases, and

² A machine-readable dictionary (MRD) is an electronic dictionary compiled for processing purposes by NLP software, and containing information, which is normally not present in the conventional dictionaries for human users.

³ A text corpus is a large and structured set of texts used for linguistic analysis.

- (ii) An *ontologically tagged corpus* with tags whose definitions are contained in this ontology dictionary.

5.2 BUILDING AN ONTOLOGY DICTIONARY

5.2.1 Selecting Dictionaries

Three features are considered important when choosing a dictionary source for building an ontology dictionary. These include: grouping by meaning, a good dictionary structure and large number of word entries.

(1) Grouping by meaning

When building an ontology dictionary, words and phrases with similar meanings or close connections should be grouped together, even if they are in different parts of speech categories. For example, “resemblance” and “similar” should be grouped together because they have similar meaning although “resemblance” is a noun, and “similar” is an adjective. “Close connections” here means words and phrases that are not similar in meaning, but could be associated with each other. For instance, “bank” and “cashpoint” are not synonyms, but they should be grouped together as having a close connection.

(2) A good dictionary structure

The structure of an ontology dictionary depends greatly on the structure of the

dictionary source used, and its structure also affects the scalability of the ontology dictionary. A good dictionary structure can reduce the cost for building an ontology dictionary and facilitate its expansion when more entries need to be added into it.

In particular, having leaf nodes with the same depth is important in terms of the dictionary structure. The *depth* of a leaf node⁴ in a graph structure is defined as the path from this leaf node to the root node [Devitt and Vogel, 2004]. Leaf nodes with different depths cause a data storage problem when building an ontology dictionary, as the memory space to store the dictionary data cannot be predetermined. This, in turn, may affect the speed of processing data contained in this ontology dictionary.

(3) Large number of word/phrase entries

Obviously, more words and phrases in the dictionary source will increase the vocabulary volume of the ontology dictionary.

5.2.2 Dictionaries

Based on the considerations discussed in 5.2.1, two dictionary sources are identified as candidates for building an ontology dictionary. These are WordNet [Miller, 1990] and Roget's Thesaurus [Roget, 2003].

⁴ A leaf node in this work refers to words/phrases in a dictionary.

WordNet

WordNet developed by Princeton University is a generic lexical reference system for the English language. Its design is inspired by current psycholinguistic theories of human lexical memory [Fellbaum, 1998]. WordNet has an acyclic graph structure allowing multiple inheritance. WordNet 1.6 contains 152,059 unique words and 203,145 word-sense⁵ pairs. WordNet is well studied and widely used: its official web site [Princeton University, 2006] contains 387 papers referring to WordNet.

*Semcor*⁶ is a package of semantic concordance text annotated using information from WordNet. A semantic concordance is a textual corpus and a dictionary combined in such a way that every word/phrase in the text is linked with its appropriate sense in the dictionary. Therefore, it can be viewed either as a corpus with syntactically and semantically tagged words, or as a dictionary containing example sentences for many different definitions [Fellbaum, 1998]. *Semcor* contains 352 text files selected from the Brown Corpus⁷, which is widely used in the field of linguistics and computer science. All words in *Semcor* are annotated in the same fashion, using tags with attribute - value pairs. The tags used in *Semcor* are listed in Table C.1 (Appendix C).

The following example illustrates the tagging in *Semcor*. The word “said” is tagged

⁵ ‘Sense’ refers to one of the meanings of a word or phrase.

⁶ The *Semcor* package used in this research was distributed with WordNet 1.6.

⁷ The Brown Corpus of Standard American English (Brown Corpus) was compiled by Henry Kucera and W. Nelson Francis at Brown University.

with five attributes (cmd, pos, lemma, wnsn, lxsns) and their corresponding values (done, VB, say, 1, 2:32:00) as shown on line 8 of Figure C.1 (Appendix C):

```
<wf cmd=done pos=VB lemma=say wnsn=1 lxsns=2:32:00::>said</wf>
```

In the entry, “cmd=done” means that the word “said” has been properly tagged; “pos=VB” indicates that it is a verb; “lemma=say” specifies its lemma; “wnsn=1” shows that the sense refers to the first sense of “say” in WordNet, and “lxsns=2:32:00::” contains the lexical sense number of the word “said” (“2:32:00”).

Roget's Thesaurus

Roget's Thesaurus (Roget's) is a thesaurus mainly used to facilitate the expression of ideas and assist in literacy composition. It has a tree structure that provides systematic classification of its vocabulary. Roget's tree structure contains eight levels as follows.

1. *Top level classes.* These are the most general classification categories for all words/phrases in Roget's. Examples of top level classes are “space”, “matter”, “abstract relations”, etc.
2. *Sections.* These are more detailed categories under top level classes. Examples include “existence”, “relation” and “quantity” which belong to the top level class “abstract relations”.
3. *Subsections.* These are subcategories of sections. For example, section “dimensions” is subdivided into three subsections: “general”, “linear” and



“central”.

4. *Head groups*. These are subcategories of subsections. For example, subsection “results of action” contains five head groups: “completion and noncompletion”, “success and failure”, “trophy”, “prosperity and adversity” and “averageness”.
5. *Concepts*. The latest edition of Roget’s Thesaurus contains more than 900 different concepts, such as “action”, “attack”, and “book”.
6. *Part of speech (POS)*. POS categories (such as nouns, adjectives, verbs, adverbs, and interjections) subdivide concept categories further. For example, “resemblance” and “similar” are in the same concept category but in different part of speech groups, where “resemblance” is in the group of nouns and “similar” is in the group of adjectives.
7. *Concept groups*. The words and phrases within each POS category are subdivided further into concept groups. Each concept group begins with a word known as its *headword*. It indicates the meaning of the words/phrases included in the concept group. For example, in the noun POS category of “giving”, there is a concept group with a headword “giver”. Examples of other words in this group are “donor”, “bestower” and “rewarder”.
8. *Groups within concept groups*. The words and phrases within each concept group are subdivided according to their difference in meaning, context or level of usage (i.e. colloquial or formal, etc.). For example, “giver”, “donor”, “bestower” are in

the same concept group with “rewarder”, “tipper” and “briber”, but in different subgroups, because the former are more general, and the latter are specifically related to money. In addition, some concept groups also contain cross-references to other parts of the thesaurus.

5.2.3 Comparisons between Roget’s Thesaurus and WordNet

The three features identified and discussed in section 5.2.1 are used to compare WordNet and Roget’s Thesaurus for the purpose of this study. The result is shown in Table 5.1.

First, WordNet uses *sensets*⁸ to categorise words and phrases of same part of speech category. Roget’s, on the other hand, maintains many well-structured interconnections between different parts of speech. For example, “wind” is a noun and “windy” is an adjective. In Roget’s they are included in the same concept “wind”, whereas “windy” can be found in two adjective synsets in WordNet, but neither of them contains the word “wind”.

Second, Roget’s uses over 900 concepts to organise the words and phrases contained in it, while WordNet employs only 15 relationships (such as hypernyms and antonyms) to express connections between words. For example, in Roget’s *concept* category ‘treasury, “bank” and “cashpoint” are in the same group although they are not

⁸ A *senset* refers to a set of synonyms to identify a meaning in WordNet.

Table 5.1 Comparisons between Roget's Thesaurus and WordNet

Features Required	Roget's Thesaurus	WordNet
Grouping by meaning	Many and well organised	Very few
A good dictionary structure	Well constructed classification which uses 900-1,000 concept classes. All words/phrases organised in nodes with the same depth. Suitable for searching	15 basic classifications. Words/phrases organised in nodes with different depths, sometimes more than 9 levels [Devitt and Vogel, 2004]
Large number of word/phrase entries	250,000 [Roget, 2003]	200,000 (WordNet 1.6) [Miller, 1990]

synonyms, whereas in WordNet, there are no means to describe such semantic relationships. Also, the fact that only one relationship of a given word/phrase can be used at a time to explore other words limits the usability of WordNet.

Furthermore, the equal depth for all the word/phrases in Roget's facilitates the building of an ontology dictionary.

Having a dictionary source with a large vocabulary is obviously an advantage in building an ontology dictionary, because more words and phrases can be included in it.

This comparison shows why Roget's Thesaurus is considered in this study a better dictionary source for building the ontology dictionary: (i) the words/phrases in Roget's are grouped by meaning, (ii) it is better structured, and (iii) it has larger number of words/phrases than WordNet. The process of building the ontology dictionary is described in the next section.

5.2.4 Building an Ontology Dictionary from Roget's Thesaurus

Two practical considerations are taken into account when creating an ontology dictionary from Roget's Thesaurus.

1. No cross-references

The existing cross-references in Roget's are not included in the ontology dictionary to

keep its structure simple.

2. No subgroups within concept groups

As discussed at the end of section 5.2.2, the words/phrases in a concept group are further separated into groups due to the small difference in their meanings or usages. Such differences are ignored in this work in order to reduce the complexity of the ontology dictionary.

The ontology dictionary, named “*OntoRo*”, is created using the electronic version of Roget’s built in Project Gutenberg [Project Gutenberg, 2005], and the printed version of Roget’s Thesaurus of English Words and Phrases [Roget, 2003]. The Gutenberg’s edition was used as the main dictionary source, while the printed 2003 edition was utilised to remove outdated word entries from Gutenberg edition, and add new entries into the ontology dictionary. The ontology dictionary created contains 68,920 unique words and 228,130 entries, which are classified into 990 *concepts*⁹, 610 *head groups*, 95 *subsections*, 39 *sections*, and 6 *top level classes*.

The entries in the ontology dictionary follow the format shown in (5.1).

word/phrase, concept, concept group, POS, head group, section, subsection, top level	(5.1)
---	-------

The *fields* in (5.1) are separated by commas. The first field contains the word/phrase

⁹ Called “heads” in some editions.

itself; the second field contains the concept number; the number in the third field indicates the concept group number; the letters in the fourth field (“n”, “v”, “a”, “adv”) represent parts of speech (noun, verb, adjective, and adverb respectively). Following that are the numbers indicating the head group, section, subsection and top level class.

After the entries of the ontology dictionary are created, the format needs to be converted into the (5.1) format by using regular expressions¹⁰. Two software tools, AWK and VIM, are used during the conversion process. AWK is a general-purpose computer language that is designed for processing text data, either in files or data streams [FSF, 2006]. VIM is an open-source, multi-platform text editor [VIM, 2005]. Regular expressions were used in AWK scripts and VIM command lines to convert the Roget’s entries into entries of the ontology dictionary.

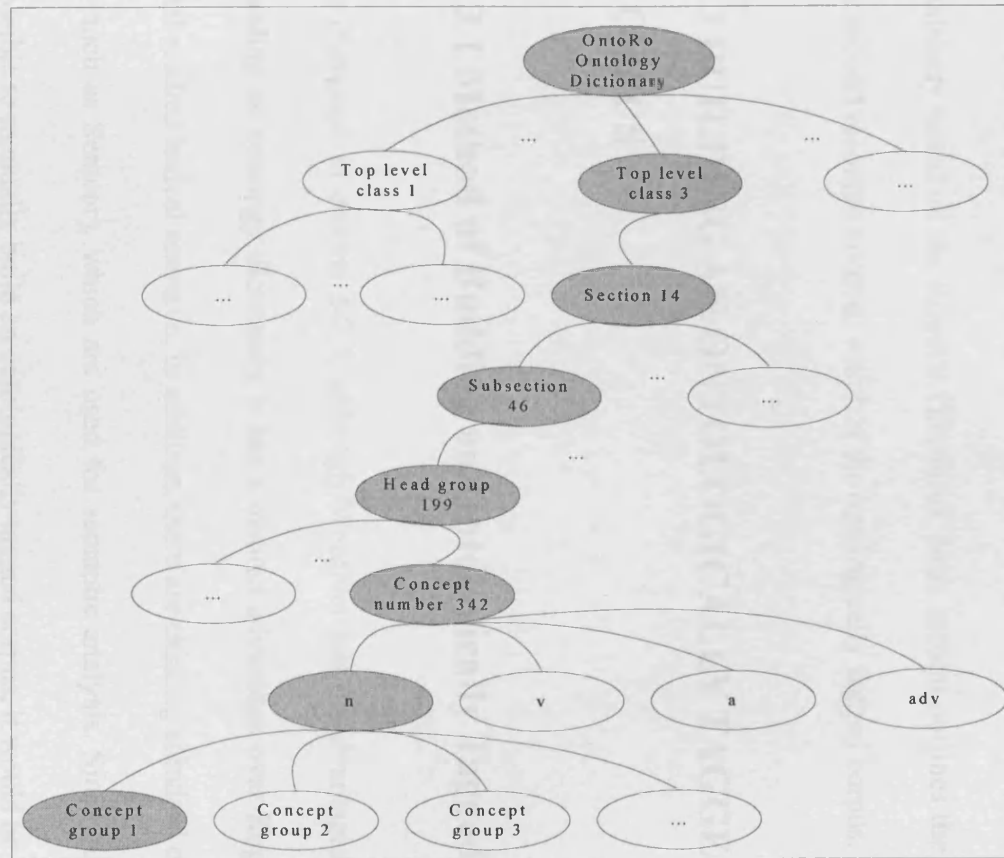
The entry for “land” in Roget’s Thesaurus [PGLAF, 2006] is shown in Figure C.2 (Appendix C); the equivalent of the same entry in the ontology dictionary is shown in Figure 5.1 (a).

The speed of retrieval is the most important factor because sequential search in such a large ontology dictionary for relevant entries is lengthy and almost impractical. Thus, a

¹⁰ A regular expression is a string that describes or matches a set of strings, according to certain syntax rules, whose origin lies in automata theory and formal language theory used in theoretical computer science.

342,n,1,land,199,46,14,3
...
342,n,2,highland,199,46,14,3
...
342,n,3,ancon,199,46,14,3
...
342,n,6,real estate,199,46,14,3
...
342,v,1,debark,199,46,14,3
342,a,1,ripicolous,199,46,14,3
...
343,adv,1,on land,199,46,14,3

(a)



(b)

Figure 5.1 A Sample Entry in the Ontology Dictionary

hashing method with $O(n)$ time complexity for searching [Witten et al., 1999] was used to improve search speed by using Java JDK's hash table function with uniform load factor of 0.75 [Joy et al., 2005]. This section described the development of an ontology dictionary based on the Roget's Thesaurus. Next section outlines the development of the second resource needed, which is the ontologically tagged corpus.

5.3 BUILDING AN ONTOLOGICALLY TAGGED CORPUS

5.3.1 Method of Building an Ontologically Tagged Corpus

As discussed in section 5.2.3, although WordNet has disadvantages with regard to building an ontology dictionary, it has a distinct advantage over Roget's for being a well-studied lexical resource. In addition, there are existing standard corpora based on it (such as Semcor), which are used for semantic analysis. Since currently it is not possible to manually build an ontologically tagged corpus, it would be a good choice in terms of time and cost involved to generate one from Semcor with the help of the ontology dictionary developed in this research.

The method for building an ontologically tagged corpus involves three steps:

Step 1. Extracting all word entries from WordNet and saving them into a machine-readable dictionary.

Step 2. Building the mapping between the WordNet entries and those in the ontology dictionary developed in section 5.2, so that each entry in WordNet is mapped to an entry in the ontology dictionary with a similar meaning.

Step 3. Converting the Semcor corpus, which is originally tagged using WordNet senses, to a corpus tagged using concepts from the ontology dictionary, by using the mapping developed in step 2.

In reality, not all entries in WordNet can be appropriately mapped to the ontology dictionary, but it is believed that inaccurate mapping entries are only a small proportion of all mapping entries generated. A hypothesis is made in this research that most of the entries in WordNet have appropriate matching entries in the ontology dictionary. The experiments described later in this chapter are conducted with the aim to test this hypothesis.

5.3.2 Building a Machine Readable Dictionary from WordNet

This section outlines the building of a machine readable dictionary called *eWord*. As the information in WordNet is stored in a specially designed database, all word/phrase entries need to be retrieved to form a machine readable dictionary as described in step 1.

An entry retrieved from WordNet is shown in Figure C.3 (Appendix C.3). The

word/phrase at the beginning of an entry is called an *entry word*, like the word “ABANDON” in Figure C.3 (Appendix C.3). The entry word “ABANDON” as a verb (“v”) has five senses. The words and phrases in each sense entry are used to express the meaning of that sense. For example, in sense 2, “abandon” and “give up” are grouped together as “abandon, give up” to differentiate this sense from other senses. In addition, each sense entry contains an explanation of its meaning and example sentences. For instance, in sense 2, the phrase “Give up with the intent of never claiming again” provides an explanation, which is followed by three example sentences.

Figure 5.2 shows the entries created in eWord from the entry of the verb “ABANDON” in WordNet (Figure C.3, Appendix C.3). The process involves several processing tasks as follows.

1. Transforming each WordNet sense into a single eWord entry. As shown in Figure C.3 (Appendix C.3), the five senses of the verb “ABANDON” in WordNet form five separate entries in eWord. This improves the machine readability and simplifies the format of eWord.
2. Deleting example sentences to improve the mapping accuracy. This will be discussed later in this chapter.
3. Removing all linking words and phrases that give little information about the entry senses. Examples include “or”, “of”, “relating to”, “refers to”, “in regard to”, etc.
4. Replacing all possessive and personal pronouns with genders such as “he”, “his”,

abandon^v^1^abandon,forsake,leave behind

abandon^v^2^abandon,give up,give up,with,intent,never,claiming,again

abandon^v^3^vacate,empty,abandon,leave behind,empty,move out

abandon^v^4^abandon,give up,give,stop,maintaining,insist on,ideas,claim

abandon^v^5^abandon,forsake,desolate,desert,lurch,leave,one,needs,count on,leave in the lurch

Figure 5.2 The Entries Created in eWord from the Entry of the Verb “ABANDON” in

WordNet 1.6

“she”, “her”, “hers”, etc. by gender-neutral, singular nominative pronouns such as “one” and “one’s”. This makes the format more consistent, and improves the mapping accuracy. eWord is built using 77,022 WordNet entries. WordNet JNI Java Native Support (WNJN) 1.1 [Bou, 2005] and regular expressions were used to facilitate the building process.

5.3.3 A Heuristic Approach for Semantic Mapping Between eWord and OntoRo

General Approach

The second step in creating an ontologically tagged corpus is to find a proper mapping for each entry in eWord to an entry in OntoRo by making use of the information in the descriptions of the entries.

However, in many cases, the problem experienced is a many-to-many mapping problem. In other words, more than one eWord entry could be mapped to an OntoRo entry and vice versa. For instance, the entry word “ABANDON” in eWord has two senses in noun form, five senses in verb form, while the same word has sixteen entries in OntoRo. In addition, other entries in OntoRo could also match the meaning of “ABANDON” in eWord. This many-to-many mapping problem is illustrated in Figure 5.3. Therefore, the problem to solve here is how to effectively find a suitable entry in OntoRo that matches every entry in eWord using only information from both

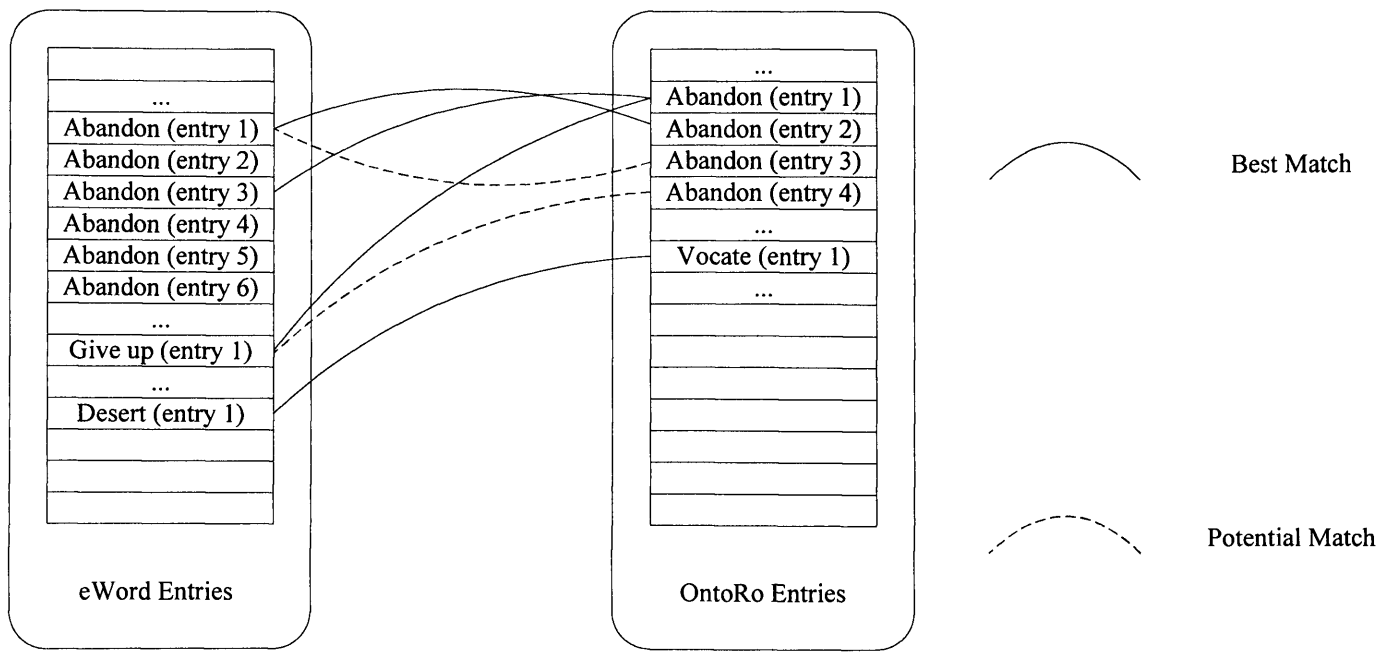


Figure 5.3 Mapping between eWord and OntoRo Entries

dictionaries. In this case, an effective mapping algorithm is needed.

The proposed mapping algorithm has three steps. The first step is to collect and calculate information needed for the consecutive steps of the algorithm. The second step is to assign an ontology tag from OntoRo to each word/phrase contained in the description of a given eWord entry. The third step involves choosing one tag among those assigned to the words/phrases in the description, which best represents the meaning of the whole eWord entry, thus creating the mapping between a given eWord entry and its corresponding OntoRo entry. The process continues until all entries in eWord have been assigned a tag from OntoRo.

The problem in the first step is that every word/phrase in an eWord entry may have several possible senses to choose from, although in context there may be only one most suitable choice. Therefore, a method is required to choose an appropriate sense from all possible senses of an eWord word/phrase. The hypothesis in (5.2) is used as a heuristic to solve this problem.

For each word/phrase in a given eWord entry description, the candidate (5.2) OntoRo sense for it is the most frequently used OntoRo sense among all words/phrases in that description.

For the whole entry, the sense candidate for representing the entry word is the most frequently used OntoRo sense among the senses assigned to words/phrases in the description.

This hypothesis is based on the observation that the words/phrases in the description of a given eWord entry often have very similar meanings. Thus, if ontology tags from OntoRo are attached to them, they should reveal a certain degree of similarity.

This approach is illustrated in Figure 5.4. In an eWord entry description, the words/phrases for the entry word E are A-D, and each of them has several possible senses in OntoRo. According to (5.2), sense No.1 for A, sense No.1 for B, sense No.1 for C and sense No. 2 are first identified as sense candidates. Next, sense No.1 is selected to represent the sense of the entry word E, as it is the most frequently used sense in this eWord entry.

This hypothesis is similar to Yarowsky's hypothesis of "one sense per discourse" for human languages, which states "with a high probability an ambiguous word has only one sense in a given collocation" [Yarowsky, 1993]. However, there are two substantial differences between these two hypotheses. First, the Yarowsky's hypothesis concerns general scenarios for free text, while hypothesis (5.2) is specific to eWord entry descriptions. The second difference is that Yarowsky's hypothesis applies to one word in a given context, while hypothesis (5.2) addresses all words/phrases in a given eWord entry.

As being a hypothesis, (5.2) does not guarantee that *most* of the words and phrases in the entry use the *same* sense. Evaluations need to be conducted to prove this hypothesis.

The details of these tests are included later in this chapter.

In addition, composition structure and linking words are two aspects which need to be considered when developing the algorithm based on this hypothesis. Both aspects are related to the pre-processing of the eWord entries before conducting the mapping

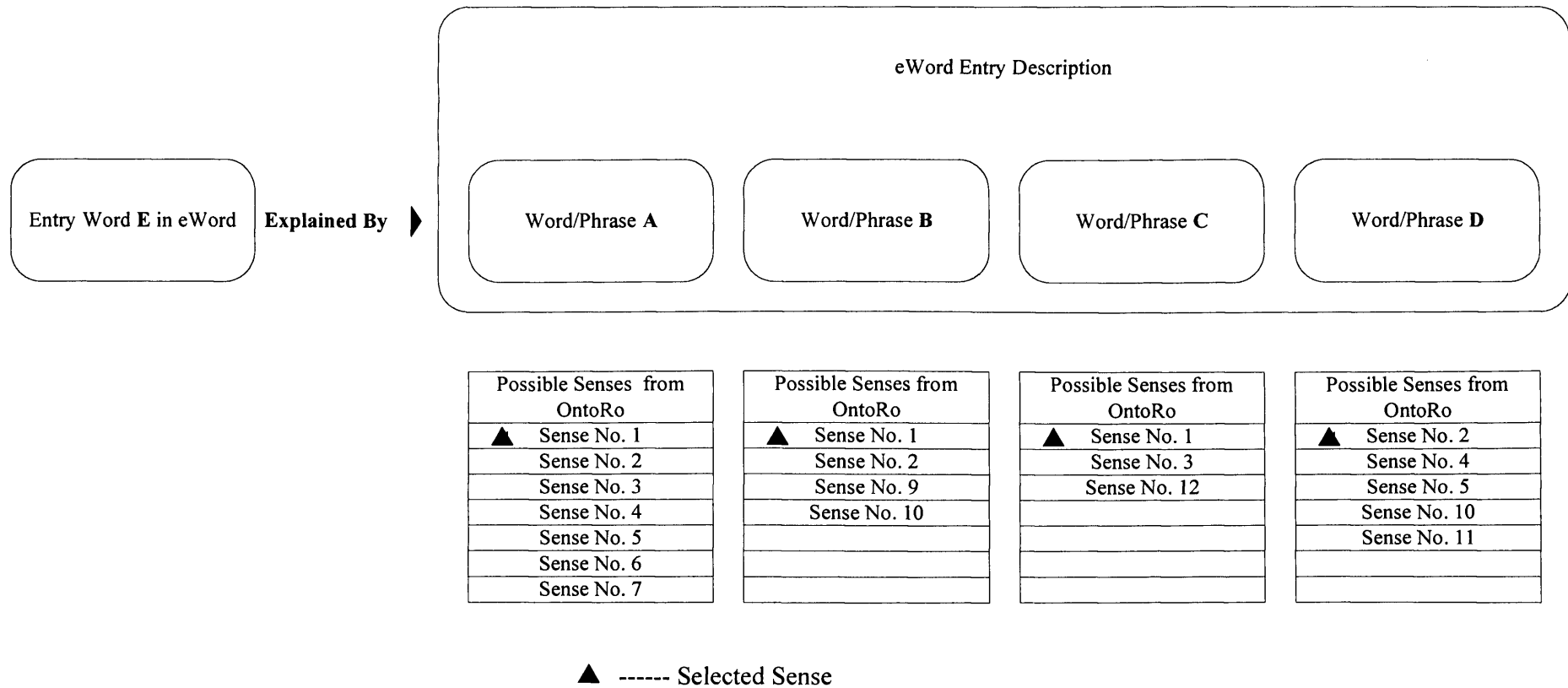


Figure 5.4 Illustration of the Approach based on the “One Sense Per eWord Entry” Hypothesis

procedure.

The first aspect of the pre-processing is the composition structure. Sometimes the entry word is described through words/phrases with entirely opposite meaning. For example, in the description of the entry word “abandon” in Figure C.3 (Appendix C.3), “give up with the intent of never claiming again”, “claiming” is used with a negation word “never” to express the meaning of “abandon”. Without being removed, these words may cause the algorithm to produce a poor mapping result.

In addition, many words in the example sentences of the entry descriptions are not directly related to the sense, but only describe the context of its use. For instance, in the example sentence “Abandon your life to God” (Figure C.3 in Appendix C.3) for the entry “abandon”, neither “your life” nor “God”, is directly related to the sense of “abandon”. These irrelevant words could also cause a poor mapping result if hypothesis (5.2) is used. Therefore, all example sentences are removed from the entry descriptions.

The second aspect of the pre-processing concerns the linking words. Normally, they do not carry important information for the corresponding entry description. Typical linking words are “with regard to”, “referring to”, “refers to”, “regarding”, “relating to”, and “or”. These words and phrases are only meaningful when they are related to “connected with” or “related with”, but the use frequency in this way is very low. For instance, the word “concerning” appears in eWord 30 times in total, but only 3 times it contributes to the entry description. Other words that are typically used as linking

words also have similar statistics. Therefore, to make use of the hypothesis (5.2) and simplify the processing, words with typical linking word usage are all removed from the eWord entries.

The mapping algorithm is shown in Figure 5.5. First, various types of information is calculated and stored, then one sense is identified as the candidate for each word/phrase in the entry, and finally one sense is selected from them as the final choice for the entire entry.

In (a), the corresponding eWord and OntoRo files are opened, the data structures are initialised and relevant memory resource is allocated. To facilitate the processing, the parts of speech indicators in the OntoRo entries are replaced by numbers (1=noun, 2=adjective, 3=verb, 4=adverb). The general form of an OntoRo entry is a sequence of numbers corresponding to: concept, concept group, POS, head group, section, subsection, and top level.

From (b) to (i), the algorithm uses several loops until all eWord entries are processed. In step (c), the algorithm scans for the longest word combination (phrase¹¹) in a given number of words in the description, that also appears in OntoRo. The word combination found is called an *element*. After one element is identified, the scan continues from the next word until all the words in the description are scanned. For example, in the eWord

¹¹ One or more words.

Initialisation, such as opening eWord and OntoRo files, etc.	(a)
For each entry in eWord	(b)
{	
Read the words from the beginning of each entry, sequentially identify the longest phrase combinations as elements, and store them for later use.	(c)
For each element in (c), find all possible senses, and store them for later use	(d)
{	
For each element count how many other elements in the same entry have the same number for concept, head group, subsection, section, and top level	(e)
{	
Order the senses according to their semantic similarity based on the counts obtained in (e). The semantic similarity is computed for each field in sequence: concept, head group, subsection, section, and top level.	(f)
}	
}	
For each element in the entry, select the top ranked concept in (f) as the concept candidate according to the semantic similarity. Store all concept candidates for later use.	(g)
For each element with the selected sense candidate in (g)	(h)
{	
Rank these elements according to their semantic similarity, choose the highest ranked element, and output the corresponding entry in OntoRo.	
If there is no such an element in (h), find the element with the lowest frequency of appearance in eWord from the first 5 elements in that entry. Find the corresponding entry in OntoRo and output the result.	(i)
}	
}	
Close all files opened, free any memory allocated.	

Figure 5.5 An Algorithm for Mapping eWord and OntoRo Entries

description “give up the career...”, both “give” and “give up” can be found in OntoRo. However “give up” is selected as an element because it contains two words as apposed to “give”. After choosing “give up”, the scan continues from the next word (“the”).

In (d), one element could appear in different entries of OntoRo with different meanings, and they should all be stored as candidates.

Next, in (e), for each field (such as concept, head group, subsection, section, and top level), the algorithm counts the number of instances when one element shares the same value as another element. The statistics obtained in (e) is further used in (f) to calculate the semantic similarity of all possible senses for each element.

Here, semantic similarity refers to the degree to which one sense is close to the meaning of another. The semantic similarity is measured by pair-wise comparisons of the values in the fields of the ontology entries, in the following order of priority: concept, subsection, section and top level. The elements are compared on a lower priority level only if the values in the field with a higher priority are the same. If two elements have the same degree of semantic similarity, then the one that appears first in the candidate list is selected.

In (f), for each element, the possible senses are ranked according to the semantic similarity calculated in (e).

In (g) for each element, the top ranked sense in (f) is selected as a sense candidate for

that element.

In (h), the senses selected in (g) are ranked according to their semantic similarity. The semantic similarity is calculated in a similar way as in (f). However, the similarity measure in (h) is used to compare the senses, while in (f) it is applied in the comparison of the possible senses for each element. Finally, the highest ranked OntoRo sense is selected to represent the given eWord entry.

This approach cannot be applied in those cases when there are no fields with the same values. In this case hypothesis (5.2) cannot be used to determine which sense should be selected as the candidate to represent the eWord entry word. Such cases are addressed in (i), where the element with the lowest occurrence in eWord is selected from the first five words in the description of that entry. This approach is based on the observation that words and phrases less frequently appearing in a text carry more information [Shannon, 1948]. The selection is limited to the first five words only, because simple tests reveal that the most appropriate sense for an entry word is often among the senses assigned to the first five words.

Figure 5.6 illustrates this algorithm by depicting the possible sense entries in OntoRo of the eWord entry shown in (5.3).

abandon ^v ³ vacate,empty,abandon,leave behind,move out	(5.3)
--	-------

In Figure 5.6, “count” represents how many times the concept number of one element is shared with the other elements. For example, the first entry of the element “abandon”,

Possible Sense Entries in OntoRo

vacate		empty		abandon		leave behind		move out	
Count	Entries	Count	Entries	Count	Entries	Count	Entries	Count	entries
1	190,1,3,112,9,33,2	0	325,1,2,189,14,45,3	2	621,1,3,371,27,62,5	1	41,2,3,23,3,11,1	2	621,1,3,371,27,62,5
0	752,1,3,455,31,70,5	0	774,2,2,471,34,73,5	0	944,1,1,577,38,89,6	0	306,1,3,178,12,43,2		
1	190,2,3,112,9,33,2	0	477,1,2,278,19,52,4	0	835,1,1,512,36,78,6	0	34,1,3,20,3,10,1		
2	621,1,3,371,27,62,5	0	172,1,2,99,8,28,1	0	943,1,1,576,38,89,6	0	285,1,3,167,12,43,2		
1	753,1,3,456,31,70,5	0	572,1,2,339,25,58,4	0	678,2,1,408,28,65,5	0	277,3,3,163,12,41,2		
		0	877,1,2,538,36,82,6	0	822,2,1,505,35,77,6	0	306,3,3,178,12,43,2		
		0	450,1,2,261,16,49,4	0	833,2,1,511,36,78,6	0	506,1,3,294,21,54,4		
		0	859,3,2,526,36,80,6	0	779,1,3,475,34,73,5				
		0	946,1,2,579,38,89,6	0	603,1,3,360,26,59,5				
		0	679,1,2,408,28,65,5	1	753,1,3,456,31,70,5				
		0	636,2,2,384,27,63,5	1	41,2,3,23,3,11,1				
		0	515,1,2,301,23,56,4	0	57,2,3,32,3,12,1				
		0	190,2,2,112,9,33,2	0	918,1,3,563,38,86,6				
		0	541,2,2,317,24,57,4	0	458,1,3,266,17,50,4				
		0	543,1,2,319,24,57,4	0	779,1,3,475,34,73,5				
					



The Sense Selected for Each Element

vacate	empty	abandon	leave behind	move out
621,1,3,371,27,62,5	325,1,2,189,14,45,3	621,1,3,371,27,62,5	41,2,3,23,3,11,1	621,1,3,371,27,62,5



vacate
621,1,3,371,27,62,5

Candidate Element and its Sense for the Whole Entry

Figure 5.6 Illustration of the Semantic Mapping Algorithm

which is “621,1,3,371,27,62,5”, shares the same concept number “621” twice with other elements (“vacate” and “move out”), thus the number in the “count” field is 2.

The mapping algorithm selects one candidate sense for each element of the entry description as shown in Figure 5.6. Finally, the candidate “vacate” for the entry word “abandon” is selected according to the algorithm (h) to (i) in Figure 5.5, because the concept number “621” appears three times and is the most frequently used sense in the entry. Thus, the eWord entry word “abandon” in (5.3) is mapped to the sense entry “621,1,3,371,27,62,5” of “vacate” in OntoRo.

Testing of the Semantic Mapping Algorithm

The algorithm is implemented in C language using Microsoft Visual Studio 6.0. Sample source code developed is included in Appendix D. The tests were conducted on a Pentium III 700MHz 384MB memory computer. The total running time for mapping all 77,022 entries in eWord to 37301 OntoRo entries was 418 minutes.

In addition, tests were carried out to examine the mapping accuracy. 200 entries were randomly selected from the 77,022 entries already mapped and were reviewed by an expert. The entries are selected at a defined interval across the whole mapping result, so that the observed error rate could be considered representative of the error rate of all 77,022 entries. The judgment criteria used by the expert are listed below.

Criterion 1	The word/phrase selected in OntoRo as a result of the mapping should be close to the meaning of the eWord entry.
Criterion 2	<p>If (1) cannot be satisfied, and the sense can only be expressed as a combination of two or more entry words, then the result should be one of them, or</p> <p>If (1) cannot be satisfied, and the sense can be expressed as a hypernyms¹²-hyponyms¹³ relation, then the hypernym should be selected as a mapping word/phrase.</p>

If the mapping result satisfies either of the two criteria, then the result is correct. If criterion 1 is not satisfied, criterion 2 is used to determine the accuracy of the result.

Criterion 1 means that the mapping result should be as specific as the sense of that eWord entry, and it should not be over specific, nor too generalised. Usually, a synonym-like¹⁴ word/phrase is used to represent the meaning of that eWord entry.

According to the mapping algorithm, the mapping word/phrase is one of the words/phrases in the eWord entry description. Sometimes, however, a synonym-like word/phrase cannot be found in the eWord entry because its meaning is close to a combination of several (usually two) words/phrases in that entry. In this case, the correct result should be one of the words/phrases from this combination, because the eWord entry can only be mapped to one OntoRo entry. Furthermore, when the correct sense is a hypernym-hyponym relation, the correct result is the hypernym rather than

¹² A hypernym is a word that is more generic than a given word.

¹³ A hyponym is a word that is more specific than a given word.

¹⁴ A synonym-like word/phrase is a word that is not necessarily a synonym, but is close enough to the meaning of the sense entry.

the hyponym.

To validate the algorithm developed, 200 samples from the mapping result were analysed. 160 of them were found correct according to the expert involved. Therefore, the mapping accuracy is 80%.

Discussion

An analysis of the errors found showed that they are approximately evenly distributed within the test result. The possible causes of error are discussed and explained below.

1. No counter part entry in OntoRo. As the composition of WordNet and OntoRo is different, it is sometimes impossible to find an appropriate mapping word/phrase in OntoRo to suit the sense of WordNet. An example which is wrongly mapped is given below.

eWord Entry	DISSOLVED ^a .dissolved,(of solid matter) reduced to a liquid form	(5.4)
The Corresponding Entry in OntoRo	solid,2,11,37,143,1,243	(5.5)

In OntoRo, there is no such word as “dissolved”, therefore the algorithm skipped this word, and chose “solid” as a resultant mapping entry. In this example, there is no alternative word apart from “dissolved” that can explain the sense of the entry word well. Due to the reason above, the mapping result is wrong.

2. Long entry description in eWord. A wrongly mapped example is shown below.

eWord Entry	PARRICIDE ⁿ . parricide, someone who kills his or her parent	(5.6)
The Corresponding Entry in OntoRo	parricide ⁿ ^{one,3,15,47,217,2,371}	(5.7)

The eWord entry (5.6) contains several words/phrase for the sense of “somebody”: “someone”, “who”, “his”, “her”, and “parent”. In this case, the mapping algorithm has chosen “someone” as the mapping result, which is more related to “somebody” because it is the most frequently used sense. Those entries with long entry descriptions, where the context of usage is described not necessarily using words or phrases with similar meaning, are likely to produce wrong mapping results, because more irrelevant words or phrases are likely to be included, and some of them may cause the algorithm to produce incorrect results.

5.3.4 Converting the Semcor Corpus into an Ontologically Tagged Corpus

Once the semantic mapping from eWord to OntoRo is completed, the Semcor can be converted into a corpus using tags from OntoRo (step 3, as outlined in section 5.3.1).

An ontologically tagged corpus (OntoCorp) thus can be created.

Three rules are applied during the conversion of the Semcor corpus. These are as follow.

1. Words/phrases with little semantic information (such as “a”, “an” and “the”) or

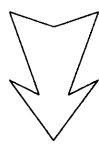
without “lemma”¹⁵ attribute in Semicor entries, as well as all proper nouns and punctuation marks are all tagged as “IGNORE”.

2. All words/phrases that cannot be found in OntoRo are tagged as “UNKNOWN”.
3. The other words/phrases are tagged using the corresponding field number in the hierarchy of OntoRo with a preceding “S” as an ontology tag. For example, “vacate” in Figure 5.6 is tagged at head group level as “S112”, and at concept level as “S190”.

A typical ontologically tagged sentence converted from Semicor is shown in Figure 5.7. In the conversion output (Figure 5.7), every word is separated with “/” from its tag. The conversion programme was implemented in Java. The Semicor corpus was converted into a text of 20,138 lines using the approach outlined above. Regular expressions and a software tool UltraEdit [IDMCS, 2006] were used to convert the Semicor text into an XML formatted file, then the file was parsed to extract the information using: the SAX Java library [ASF, 2006], the mapping file generated in section 5.3.3 and OntoRo implemented in section 5.2.4. The ontologically tagged corpus OntoCorp contains 434,998 annotated words in total.

¹⁵ Lemma refers to the canonical form of a word, usually the base form, taken as being representative of all the various forms of a morphological paradigm [Mitkov, 2003].

```
<s snum="28">
<wf cmd="ignore" pos="PRPS">His</wf>
<wf cmd="done" pos="NN" lemma="petition" wnsn="1"
lexsn="1:10:00::">petition</wf>
<wf cmd="done" pos="VB" lemma="charge" wnsn="6"
lexsn="2:32:00::">charged</wf>
<wf cmd="done" pos="JJ" lemma="mental" wnsn="2"
lexsn="3:01:00::">mental</wf>
<wf cmd="done" pos="NN" lemma="cruelty" wnsn="2"
lexsn="1:12:00::">cruelty</wf>
<punc>.</punc>
</s>
```



```
His/IGNORE petition/S312 charged/S497 mental/S260 cruelty/S556 ./IGNORE
```

Figure 5.7 A Converted Sentence with Ontology Tags

5.4 SUMMARY

In this chapter, ontology tagging is proposed to enable text to be searched, browsed and analysed at different abstraction levels, so that existing text-based document collections can be used in Semantic Web applications. An ontology dictionary and an ontologically tagged corpus are identified as two resources enabling automatic ontology tagging. Then, an ontology dictionary *OntoRo* is created from *Roget's Thesaurus*. Next, a machine readable dictionary called *eWord* is built from *WordNet* using the semantic mapping algorithm proposed. The tests conducted showed that the accuracy of this algorithm is 80%. Furthermore, an ontologically tagged corpus *OntoCorp* is built using *eWord*, *OntoRo* and *Semcor*, a package of semantic concordance text. The developed *OntoRo* and *OntoCorp* will be used to annotate free text as described in Chapter 6.

CHAPTER 6. FULL TEXT ONTOLOGY TAGGING BASED ON MACHINE LEARNING

In this chapter, a rule-based supervised machine learning algorithm for ontology tagging is developed and tested. It obtains statistical and context information from the ontologically tagged corpus (OntoCorp) and use it to produce statistical rules and context rules. In the tagging process, these rules are applied to unknown text so that an ontology tag is assigned to each word/phrase in the text. In addition, three case studies are conducted in order to optimise the design of the algorithm. Finally, the options to improve the tagging accuracy and the reasons that cause tagging errors are analysed and discussed.

6.1 ONTOLOGY TAGGING ALGORITHM

6.1.1 Statistical and Context Information

Similar to other supervised learning algorithms, training material need to be provided to the ontology tagging algorithm so that the knowledge from the material can be applied in tagging unseen text input. Therefore, it is important to present the training material in an appropriate way to the algorithm, so that the knowledge learned from the training material could be used in a way guaranteeing high tagging accuracy. For the algorithm discussed in this chapter, the training material is the ontologically

tagged corpus, and the knowledge refers to the statistical and context rules generated during the training process.

Two types of information can be obtained from the ontologically tagged corpus. These are statistical and context information, which can be expressed through eight parameters as shown in Table 6.1.

1. Statistical information from words and phrases in the corpus

Statistics has been used in many study areas of natural language processing [Zipf, 1949, Samuelsson, 2003]. It is therefore expected that statistical information (such as frequency of words, phrases and ontology tags) can benefit ontology tagging.

Four parameters, N , $F(w)$, $F(c)$ and $O(w)$, are employed as statistical measures in this study (Table 6.1). The total number of words/phrases in a given text (N) is normally used in conjunction with the other parameters. $F(w)$ refers to the number of instances a certain word/phrase w appears in a given text. $F(c)$ is the number of instances a context word/phrase c appears in the context of the word/phrase w . $O(w)$ refers to the total number of times that a given word/phrase w is annotated with an ontology tag.

2. Context information about words, phrases, ontology tags and their combinations

The remaining four parameters introduced, $F(w,c)$, $R(w,c)$, $CO(w,t)$ and $MI(w,c)$, are based on using context information (Table 6.1).

A context of a word refers to the part of a text that surrounds that word. Those words

Table 6.1 Statistical and Context Information

	Symbol	Parameters Description
Statistical Information	N	Total number of words/phrases in a given text
	$F(w)$	Word frequency for a given word/phrase w
	$F(c)$	Word frequency for a given context word/phrase c
	$O(w)$	Ontology tag frequency for a given word/phrase w
Context Information	$F(w,c)$	Word co-occurrence frequency for a given word/phrase w with a context word/phrase c in a certain context window width
	$R(w,c) = F(w,c) / F(w)$	Ratio $R(w,c)$
	$CO(w,t)$	Ontology tag t co-occurrence frequency with a given word/phrase w in a certain context window width
	$MI(w,c) = -\log(F(w,c)/(F(w)*F(c)))$	Mutual information for a given word/phrase w with one of its context words/phrases c in a given context window width

which appear within a certain distance from the word w are called *context words* of word w . If a text is treated as a long input sequence of words (including paragraph separators), then the *context window* of a given word comprises words on both sides of this word. The *width of a context window* of a word (abbr. as the *window width*) is the total number of words in this context window.

Usually words have different meanings in different contexts; this especially applies to polysemies¹ and homonyms². For example, “foot” in the context of (6.1) refers to the lowest part of an object, while “foot” in the context of (6.2) means the bottom part of the leg of a vertebrate.

The house is at the foot of the mountains.	(6.1)
One of his shoes felt too tight for his foot.	(6.2)

Therefore, it is expected that the information contained in the context can help determine the meaning of a word in a given context.

For a given word w and its context word c , the parameter $F(w,c)$ denotes the number of instances both words appear in a given *window width*. $R(w,c)$ measures how often a given word w appears together with a context word c . $CO(w,t)$ shows how often an ontology tag t appears within a given *window width* of the given word w . $MI(w,c)$, which is derived from mutual information measurement [Samuelsson, 2003],

¹ Polysemy refers a word having or being characterised by many meanings.

² Homonyms refer to words that have the same sound and often the same spelling but differ in meaning and origins.

measures to what degree one element is representative of another. In other words, $MI(w,c)$ is used to evaluate how related one word is to another. If the information about the word c is known, the information about the word w is unknown and two words are known to share a high $MI(w,c)$ value, then the information about the word c can be used to represent that of the word w .

6.1.2 Variables

Several variables that will be used later in this chapter are defined in this section. The difference between parameters and variables is that the parameters are calculated from the training material provided, while the variables can be adjusted.

1. The *ratio* S indicates, given a certain amount of material, how much of the material is used for training and how much is used for testing the algorithm. The ratio S is defined in (6.3).

$S = (\text{the number of sentences in the training sets}) / (\text{the number of sentences in the testing set})$	(6.3)
---	-------

This ratio could be used to evaluate the consequences of increasing the amount of training material and decreasing the testing material (provided that the total amount is sufficient for thorough testing).

2. The *range* based on Zipf's Law can be used as a criterion for filtering the context words in a given *window width*. According to Zipf's Law [Zipf, 1949], the frequency

of the n th-most-frequently-used word is approximately inversely proportional to n . Figure 6.1 illustrates this statement by showing an example involving a total of 100 unique words used in a text. In this example, the most frequently used word (position 100 on the horizontal axis) is mentioned 1000 times in the text. If another word is known to be the third most frequently used word in this sample (position 998), then according to the Zipf's law, it is used approximately 333 ($1000/3$) times in the text. Figure 6.2 represents the same law, with the horizontal axis been normalised for values between 0 and 1, with the most frequently used word placed at the end of the horizontal axis. The *Zipf's range* refers to a range of X-axis values representing a group of words ordered according to the frequency of their use, as shown in Figure 6.2. The range contains the words which are taken into consideration when extracting context information from the context window. This variable will be used to find out how the Zipf's law affects the ontology tagging accuracy.

3. The ratio θ is defined by (6.4).

$\theta = \varepsilon/\zeta, \quad (\zeta > 0)$	(6.4)
---	-------

where,

ε is the frequency of a context word/phrase c occurring with a word/phrase a tagged with an ontology tag b in a given *window width*, and ζ is the frequency of a context word/phrase c occurring with the word/phrase a which is tagged with an ontology tag other than b .

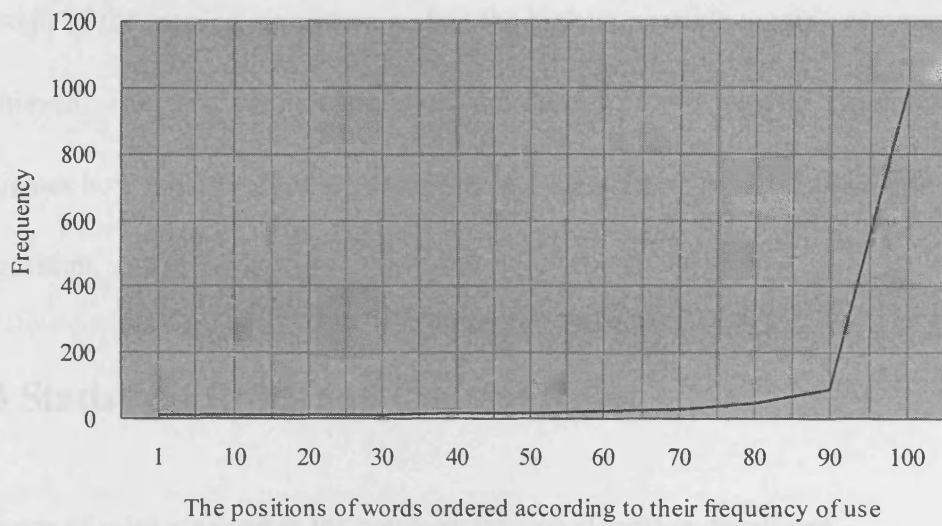


Figure 6.1 Zipf's Law

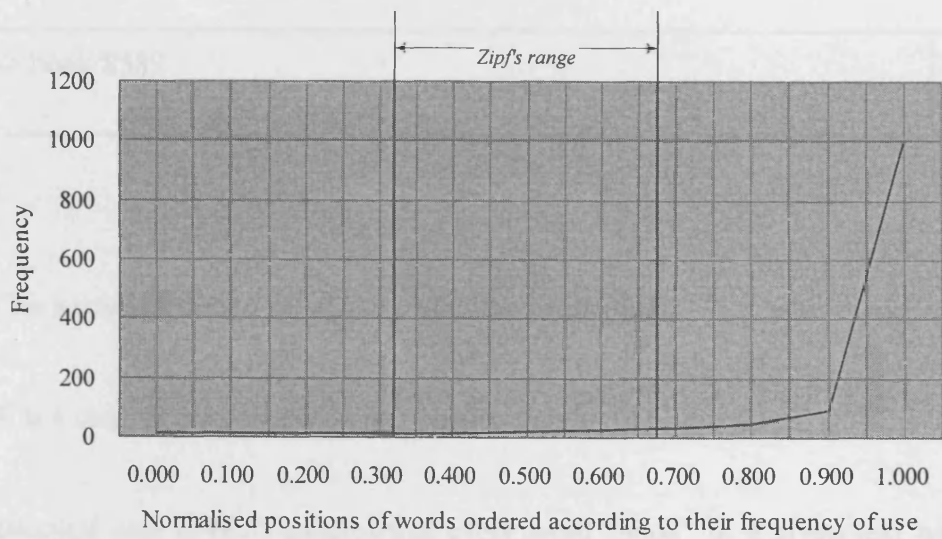


Figure 6.2 Normalised Zipf's Range

Thus the problem to solve is how to effectively use these parameters and variables in the design of the tagging algorithm, so that the highest possible tagging accuracy can be achieved. The next section discusses the design of the tagging algorithm and investigates how the use of these parameters and variables may affect the accuracy of the algorithm.

6.1.3 Statistical Rules and Context Rules

Two types of rules are used in the ontology tagging algorithm developed.

Statistical Rules

An example of a statistical rule is given below.

book -> book/S589	(6.5)
-------------------	-------

where,

“book” is a word/phrase to be tagged by the algorithm, and

“S589” is a concept tag for “books and publications”.

The statistical rule in (6.5) denotes that every word “book” in a given text will be assigned a concept tag “S589”, which is the concept related to books and publications.

As statistical rules do not use context information, such ontology tags will be attached to the corresponding words regardless of whether the concepts fit the context or not.

Therefore, a statistical rule is more effective if a given word has a dominant meaning which can be expressed through the corresponding ontology tag.

Context Rules

Two examples of context rules are shown below.

accounting, book/S589, context windows width = 4 -> book/S808	(6.6)
S586, book/S589, context <i>window width</i> = 4 -> book/S586	(6.7)

In addition to the concept tag “S589” mentioned above, these context rules use “S808” for “accountancy and book-keeping”, and “S586” which is related to “writing”.

Context rule (6.6) means that if the word “book” has an ontology tag “S589” attached, and the word “accounting” appears within a context window of four words around “book/S589”, then tag “S589” should be replaced by “S808”.

Context rule (6.7) means that if the word “book” has an ontology tag “S589” assigned, and the ontology tag “S586” appears within a context window of four words around “books/S589”, then tag “S589” should be changed to “S586”.

6.1.4 Training and Tagging with the Ontology Tagging Algorithm

The tagging algorithm proposed makes use of the two types of information discussed in section 6.1.1 in both the training and tagging processes. The flowcharts shown in Figure 6.3 illustrate this algorithm.

Training

In the training process, first statistical information and context information are obtained from the ontologically tagged corpus (OntoCorp) discussed in Chapter 5. All tags are removed from OntoCorp to create an *untagged corpus* (UC). This corpus is used as a comparison text, so that when statistical and context rules are applied in sequence to it, the tags generated can be compared with those in OntoCorp. The percentage of correctly assigned tags will be used to estimate the accuracy of the ontology algorithm.

The next step involves selecting the most frequently used ontology tag for each unique word/phrase in OntoCorp and generating *statistical rules*. These rules are used to tag the corresponding words/phrases in the UC. Then, the incorrectly tagged words are identified by comparing the tagged UC with OntoCorp. Next, the corresponding context information from OntoCorp for those wrongly tagged UC words/phrases is to generate *context rules*. These *context rules* are then used to replace some of the

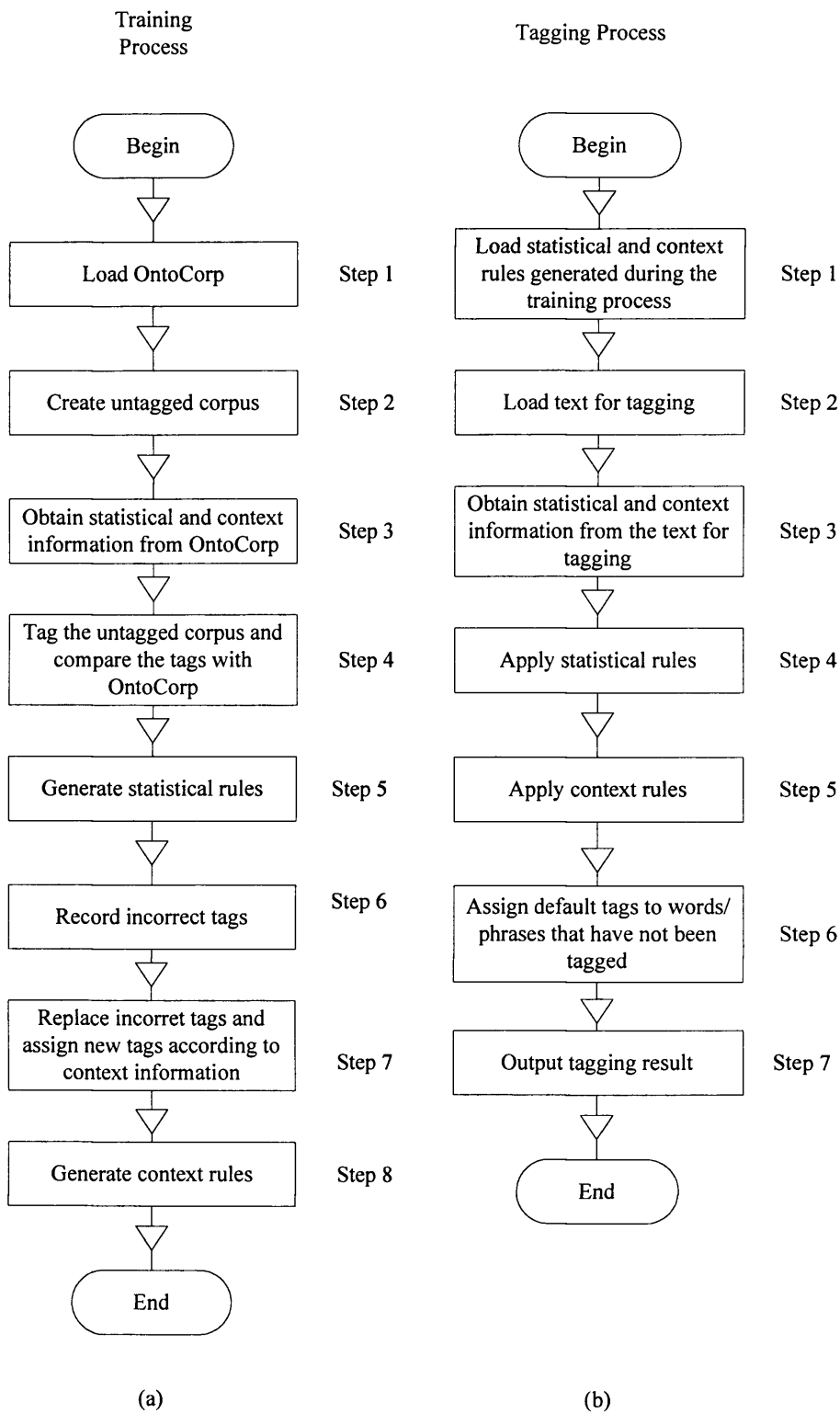


Figure 6.3 Flowcharts of the Ontology Tagging Algorithm (Training and Tagging)

attached tags and assign tags to the words/phrases which have not been tagged.

When context rules are used to replace tags, which are previously assigned according to the statistical rules, some of the correct tags may be replaced by wrong ones. Therefore, the algorithm has to ensure that when applying context rules to the partially tagged corpus, the context rules are effective enough in a sense that more wrong tags are replaced by correct ones than correct tags replaced by wrong ones. Due to this reason, the method of generating context rules has to be carefully designed and evaluated.

Tagging

In the tagging process, the statistical and context rules generated during the training are first loaded. Next, the text is processed and statistical information is obtained. Then, the statistical rules are used to assign ontology tags to the words in the text. After that, context rules are applied to replace some of the previously assigned tags or to assign tags to those words/phrases which have not been tagged. Finally, those words/phrases which neither statistical rules nor context rules have applied to, are annotated with a default ontology tag.

There are three possible design options for generating context rules. These are using: (i) word co-occurrence frequency, (ii) ontology tag co-occurrence frequency, and (iii) mutual information for words in a context window. To evaluate these design options

three case studies are conducted. They are discussed in section 6.3.

6.2 CASE STUDIES

6.2.1 General Approach

As described in section 6.1.4, there are three design options for generating context rules. These are based on using three types of context information: $R(w,c)$, $CO(w,t)$ and $MI(w,c)$ (Table 6.1). These design options are evaluated through case studies, which investigate the co-relation between the tagging accuracy and the design alternative chosen in the implementation of the algorithm.

In each case study, context rules are generated in a different way. This requires modifications in both training and tagging processes. Generally, in the training (Figure 6.3 a), different type of context information is collected to generate context rules (step 3) and different methods are used to produce context rules (step 8). During the tagging (Figure 6.3 b), different context rules are input to the algorithm (step 1), and the information from the text for tagging is collected differently (step 3). Therefore, steps 3 and 8 of the training process, and steps 3 and 1 (5 for case study 3) of the tagging process are described in more detail in the individual case studies (Sections 6.2.2-6.2.7).

In all three case studies, the *training* and *testing sets* are composed by randomly

selecting sentences from OntoCorp, which consists of 20,138 sentences with 434,998 words, and preserving a pre-defined ratio between the number of sentences in a training set and that of the corresponding testing set. Five different *ratio S* are used in the experiments (5:5, 6:4, 7:3, 8:2 and 9:1). For example, if the ratio *S* is “5:5”, then the *training set* contains 50% of all entries, all randomly selected, and the remaining 50% entries are used in the *testing set*. The purpose of using different ratios is to test the impact of the size of the training set upon the tagging accuracy³. Next, all tags are removed from the testing set to form a new set which is then processed by the tagging algorithm. The tagging accuracy is calculated by comparing the tags for each word in the new set with the tag assigned for the same word/phrase in the testing set.

The experiments conducted are summarised in Table 6.2. The 96 experiments investigate the co-relation between four variables (ratio *S*, Zipf’s range, ratio θ and window width) and three different options based on different context information ($R(w,c)$, $CO(w,t)$, $MI(w,c)$). Each experiment is repeated 6 times, and the average accuracy obtained is shown in Tables 6.3-6.8. A tagging result is considered correct if:

³ Note the assumption made that the number of sentences in a testing set is sufficient for evaluating the tagging accuracy.

Table 6.2 Summary of Tests Conducted

Case Study	Context Information Used	Variables			
		Ratio S	Zipf's Range	Ratio θ	Window Width
		Number of Experiments ⁴ Conducted			
1	$R(w,c)$	1	10	N/A	5
2	$CO(w,t)$	5	N/A	10	N/A
3	$MI(w,c)$	30	5	N/A	30

⁴ Each experiment is repeated 6 times.

(i) The tag in OntoCorp is not “unknown”⁵ and the tag produced by the ontology tagging algorithm is the same as the tag in OntoCorp, or

(ii) The correct tag is “unknown”. In this case any tag assigned by the tagging algorithm is marked as correct.

The case studies are implemented in Java. Sample source code developed is included in Appendix E. Java hashing function is used to reduce the processing time of the search procedures in the algorithms. As a result, the processing time for the whole training and tagging cycle is reduced from about 4 hours when sequential search is used to approximately 10 minutes. The tests are conducted on a computer with a Pentium 4 2400MHz CPU and 1024MB RAM and a Java 1.4.2 runtime environment.

6.2.2 Case Study 1

The training procedure employed follows the flowchart shown in Figure 6.3 (a) with modifications in step 3 and 8. Step 3 is as follows: for a given *window width*⁶, store all context words/phrases c for a given word/phrase w ; store $F(w, c)$; for this word/phrase, calculate the ratio $R(w, c)$ with every context word/phrase c , and store all $R(w, c)$. Step 8 is as follows: if $R(w, c)$ exceeds a given value, then a context rule is generated based on it.

⁵ “Unknown” tags are originally assigned by linguistic experts in the process of corpus development to indicate that there is no definite answer.

⁶ Here, the window width refers to the number of words/phrases considered on each side of the given word.

Tests are conducted to examine if the use of different *Zipf's ranges* (such as 0.00-0.10, 0.11-0.20, etc.) affects the tagging accuracy. In this case, constant window width and ratio S are used. The results (Table 6.3) show very small variations in the average accuracy, the highest (76.76%) recorded when only the words having normalised positions between 0.11 and 0.20 (Figure 6.2) are employed to extract context information.

Furthermore, with a given *Zipf's range* from 0.10 - 0.90 and Ratio S 6:4, tests are conducted to investigate how different *window widths* affect the tagging accuracy. The results (Table 6.4) show again very small variations, with better tagging accuracy achieved when relatively small context windows are used. This result was expected as normally context words/phrases close to a given word/phrase tend to have close semantic relationships and can therefore provide more important information for ontology tagging.

6.2.3 Case Study 2

In this case study, step 3 of the training process (Figure 6.3) is as follows: for a given window width, store all context words/phrases c for a given word/phrase w , and store $CO(w,t)$. The corresponding context rules are generated in step 8.

Step 3 of the tagging process is as follows: collect $CO(w,t)$ information for every word/phrase w in text, and find all ontology tags t which (1) are in the word/phrase

Table 6.3 Average Accuracy in Case Study 1: Experiments with Different Zipf's

Zipf's Range	0.00 - 0.10	0.11- 0.20	0.21 - 0.30	0.31 - 0.40	0.41 - 0.50	0.51 - 0.60	0.61 - 0.70	0.71 - 0.80	0.81 - 0.90	0.91 - 1.00
Average Accuracy, %	76.51	76.76	76.75	76.74	76.74	76.74	76.74	76.74	76.74	76.74

Ranges in Context Window Width = 12, Ratio S = 6:4

Table 6.4 Average Accuracy in Case Study 1: Experiments with Different Window

Window Width, words	10	8	6	4	2
Average Accuracy, %	76.77	76.78	76.79	76.80	76.80

Widths in 0.10 < Zipf's range < 0.90, Ratio S=6:4

w 's context window; (2) appear in one of the context rules generated for the word/phrase w ; and (3) have $CO(w,t)$ exceeding a certain threshold. If an ontology tag t satisfies all these three conditions, then it needs to be replaced by a new tag as defined by the context rule.

Experiments are conducted to test the effect of different ratios S and thresholds θ . The result of the first set of experiments for case study 2 conducted with different training ratios S is shown in Table 6.5. Clearly, the average accuracy increases from 76.40% to 77.50% as more training material is provided to the tagging algorithm. The second set of experiments study the effect of the threshold θ (Table 6.6) when the ratio S is constant. Changing the value of θ could affect the number of context rules employed as a new rule is generated when θ exceeds a certain threshold. Larger values of θ may improve the accuracy, as more incorrect tags would be replaced by correct tags. At the same time, with large θ , fewer rules would be generated as fewer words/phrases would exceed the threshold, which may have a negative effect on the accuracy. Thus, the overall improvement could be insignificant, as not many incorrect tags would be eventually changed. This hypothesis is confirmed by the results shown in Table 6.6.

6.2.4 Case Study 3

Step 3 of the training process is as follows: store $F(w)$ for all words/phrases; for a given *window width*, store all context words/phrases c for a given word/phrase w ;

Table 6.5 Average Accuracy in Case Study 2: Experiments with Different Ratio S

$$\theta=2.0, \varepsilon=20, \zeta=10$$

Ratio S	5:5	6:4	7:3	8:2	9:1
Average Accuracy, %	76.40	76.83	77.17	77.33	77.50

Table 6.6 Average Accuracy in Case Study 2: Experiments with Different Threshold

Values

Ratio $S=6:4$

Threshold θ	2.0	2.5	3.0	3.5	4.0
Average Accuracy, %	76.83	76.83	76.83	76.84	76.84

store $F(w, c)$, N , and calculate $MI(w, c)$.

Step 5 of the tagging process is: for every word/phrase find out if there is a context rule for it, and retrieve the corresponding $MI(w, c)$; if the $MI(w, c)$ exceeds a certain value, replace the corresponding ontology tag according to the context rule.

Table 6.7 shows the experimental results when different Zipf ranges are used. The test with the Zipf's range 0.12 - 0.80 produces the highest accuracy in this experiment (76.79%). The average accuracy for all other ranges remains 76.74%. This shows that Zipf's ranges 0.12-0.22 and 0.55-0.80 have effect on the tagging accuracy.

Table 6.8 shows an interesting trend in the tagging accuracy for a given fixed ratio S . The average accuracy changes as the *window width* changes. However, large *window width* does not guarantee high accuracy.

The practical implication of this result is that it is enough to process a relatively small context window of 6-8 words and still achieve a relatively good accuracy. This is because of the trade-off of large context windows, as when the context window width increases, more useful as well as more irrelevant information is also included. For example, the highest accuracy (78.91%) in case study 3, and indeed in all experiments conducted, was achieved with *window width* 6 and *ratio* S 9:1.

Table 6.7 Average Accuracy in Case Study 3: Experiments with Different Zipf's

Ranges

Zipf's Range	0.12 - 0.80	0.22 - 0.70	0.30 - 0.70	0.40 - 0.60	0.45 - 0.55
Average Accuracy, %	76.79	76.74	76.74	76.74	76.74

Table 6.8 Average Accuracy in Case Study 3: Experiments with Different Window

Widths and Ratios S

Ratio	Window Width, words						Average Accuracy, %
	2	4	6	8	10	12	
5:5	78.65	78.67	78.72	78.74	78.72	78.67	78.69
6:4	78.75	78.79	78.83	78.85	78.82	78.74	78.80
7:3	78.66	78.69	78.75	78.77	78.75	78.67	78.71
8:2	78.65	78.67	78.72	78.74	78.72	78.67	78.81
9:1	78.83	78.86	78.91	78.90	78.81	78.68	78.83

6.2.5 Discussion

The experiments within the three case studies conducted show that the average accuracy ranges from 76.40% to 78.91%, with highest accuracy achieved when the mutual information (Case Study 3) is employed. In addition, the experiments show that the provision of more training material has a positive effect on the tagging accuracy. The highest tagging accuracy achieved is 78.91% (Case Study 3), when 90% of the corpus is used for training and the context window contains 6 words.

The ontology tagging algorithm sequentially employs statistical and context rules to assign appropriate tags to the words/phrases. The statistical rules are based on the frequency of use of certain words/phrases, and their composition is more straightforward. The context rules, however, may be created in several different ways. This is in the centre of the three case studies conducted which all explore the mechanism for generating context rules. The results show a variation of about 2.5% in the tagging accuracy. The experiments also show that the statistical rules are very important for the tagging process. Similar to Yarowsky's "one sense per collocation" [Yarowsky, 1993], this approach can be called "One Ontology per Collocation". Thus the ontology tagging problem can be redefined as how to compose the context rules to be applied to the text after the tagging based on statistical rules is completed. Each case study explores that use of different variables and their effect on the tagging accuracy.

It should be noted that the accuracy is calculated using the corpus automatically generated by mapping the eWord entries into OntoRo entries. It was shown in Chapter 5 section 5.5, that the mapping accuracy is 80%. This means that the training material is not completely accurate, which brings tagging errors into the system. Therefore, the tested tagging accuracy can only be considered as an indication of the tagging accuracy. Currently, due to time constraints, it is not possible to conduct full manual corrections in the mapping of the 77,022 eWord entries into OntoRo. Moreover, once the corrections are made, all tests in this chapter should be conducted again. The results are expected to show a certain degree of variance from the results reported in this thesis. However, the results should not be very different, as nearly 80% of the training corpus is correct.

6.3 SUMMARY

In this chapter, a rule-based supervised machine learning algorithm for ontology tagging is developed and tested. Three case studies are conducted to compare the options for generating context rules in terms of the tagging accuracy they provide. The highest tagging accuracy achieved is 78.91%. The characteristics of the tagging algorithm and the factors affecting its accuracy are analysed and discussed.

CHAPTER 7. KNOWLEDGE MANAGEMENT SYSTEM BASED ON ENTITY AND CONCEPT INDEXING

In Chapter 3, the framework and methods for automatic analysis on entity and concept information are introduced and implemented. However, the method to extract entity and concept information has not been described. Furthermore, although concept information is added into the text collection through ontology tagging as described in Chapters 5 and 6, given a large text-based document collection, once entity and concept information is added into the document collection, there is no existing way to extract them from the document collection again. Therefore a method for extracting, indexing and retrieving entity and concept information is needed.

In this chapter, a conceptual model of a knowledge management system is proposed, and its system architecture for entity and concept-based knowledge management is presented and implemented. Tests on retrieval speed and computer resource consumption are conducted. Finally, the advantages of using this architecture for knowledge management are discussed.

7.1 ENTITY AND CONCEPT INDEXING

7.1.1 Conceptual Model

For large text-based systems, the conventional approach to extract information or knowledge from a system is first to *index*, then use the indexing information for further processing. The indexing technology enables fast retrieval of pertinent

information from text collections. This is a very important factor to consider when dealing with large text-based document collections. Without the indexing to support fast search and retrieval, the application of information and knowledge extraction system would not be possible in industrial practice. However, there is no existing indexing system that is suitable for the knowledge extraction task defined in this thesis.

Indexing methods based on keywords are well studied in the last decades as reviewed in section 2.2.4 in Chapter 2. Therefore, keyword indexing techniques are used in concept indexing to facilitate knowledge management in concept and entity extraction.

The system developed (Figure 7.1) has four layers: *tagging layer*, *indexing layer*, *extraction layer* and *merging layer*. The task of the *tagging layer* is to add grammatical and concept tags to the text documents, and mark the documents with unique document numbers. The task of the *indexing layer* is to provide indexing functions, so that special data structures are generated and data processing algorithms are used to improve the retrieval speed, reduce the consumption of computing resources and facilitate the extraction tasks of the next layer. Therefore, this layer is the most important layer in the conceptual model. The task of the *extraction layer* is to extract entity and concept information from a large text document collection that has been indexed. The task of the *merging layer* is to merge entity and concept information into a merged index so that entity and concept information is organised using documents as units to facilitate the retrieval task.

The conceptual model shown in Figure 7.1 is based on the conceptual framework (Figure 3.3) discussed in Chapter 3. Entity tagging, which will be discussed later in

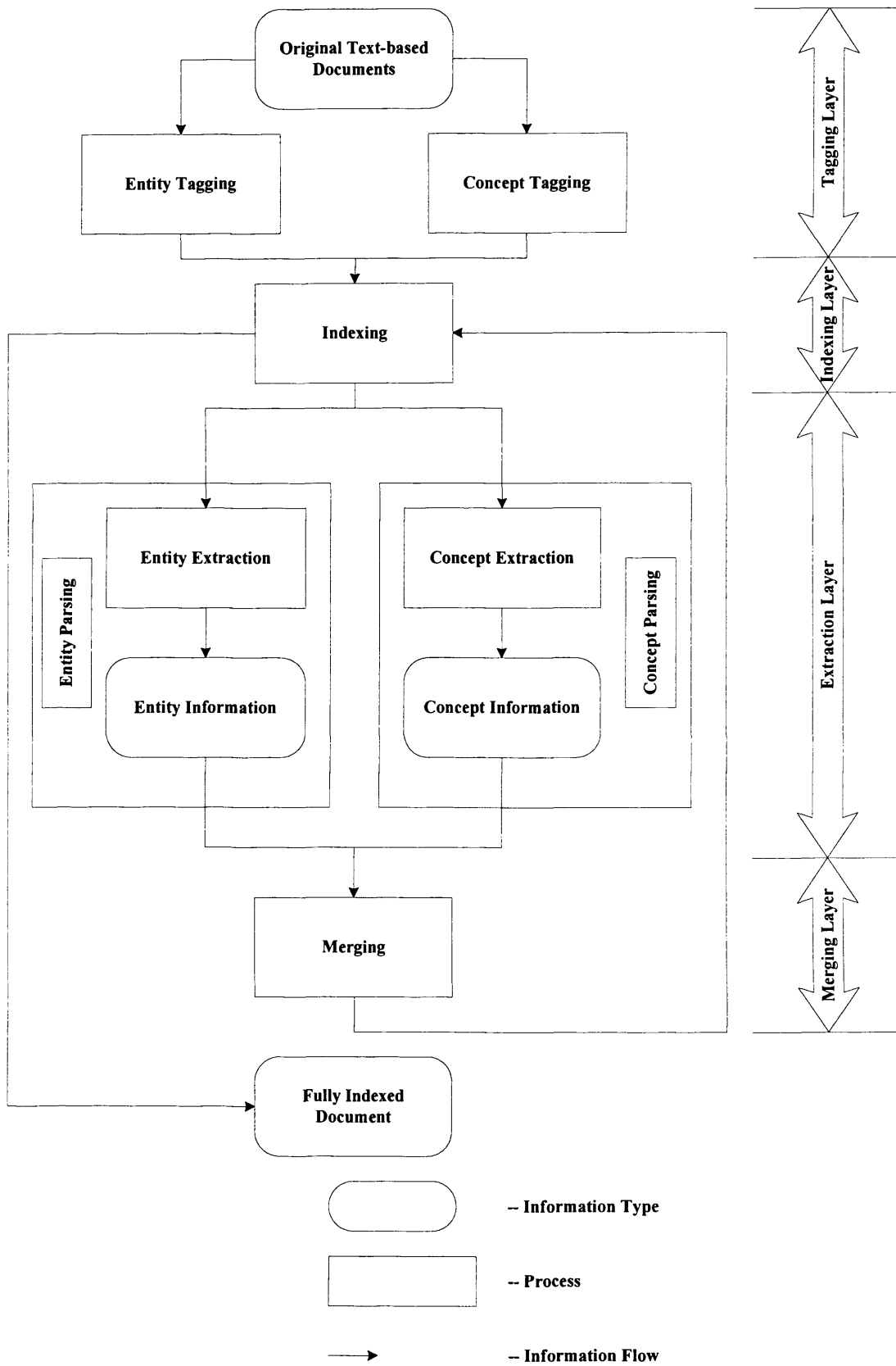


Figure 7.1. Conceptual Model of the Knowledge Management System

this chapter, involves grammar learning and syntactic analysis. Concept tagging includes concept learning and morphological analysis as outlined in Chapter 3. Concept tagging is further elaborated as ontology tagging in Chapters 5 and 6. This chapter will describe in more detail the entity tagging conducted at the tagging layer.

7.1.2 Entity Tagging

As described in section 3.1.3 of Chapter 3, an entity is an identifiable and discrete instance existing in a text document. Grammatically speaking, an *entity* refers to a word or a nominal composition of words in a sentence, and all words in this composition should be nouns or pronouns, not separated by words with other parts of speech. For example, in the phrase “paper cutter”, “paper”, “cutter” and “paper cutter” can all be treated as nominal word/phrases, but only “paper cutter” is considered an entity. On the other hand, words which are identified as *entities* can also convey ontology information. For example, “paper” in the phrase “paper cutter” indicates the concept “material”, and “cutter” relates to the concept “machine”. This type of information can be expressed by attaching ontology tags to the words/phrases. Therefore, the part of speech tagging, which assigns part of speech tags to words in the text could help to identify entities. In this thesis, this type of tagging is called *entity tagging*. The entity tagging is illustrated in the example below. The original untagged sentence (7.1) is assigned parts of speech tags (7.2), and then entities are extracted from the words that are tagged as proper noun or part of name phrase (NP), singular or mass noun (NN) and plural noun (NNS) (7.3).

The September-October term jury had been charged by Fulton Superior Court Judge Durwood Pye to investigate reports of possible irregularities in the hard-fought primary which was won by Mayor-nominate Ivan Allen Jr..	(7.1)
The/AT September-October/NP term/NN jury/NN had/HVD been/BEN charged/VBN by/IN Fulton/NP Superior/JJ Court/NN Judge/NN Durwood/NP Pye/NP to/TO investigate/VB reports/NNS of/IN possible/JJ irregularities/NNS in/IN the/AT hard-fought/JJ primary/NN which/WDT was/BEDZ won/VBN by/IN Mayor-nominate/NN Ivan/NP Allen/NP Jr./NP ./.	(7.2)
September-October term jury, Fulton, Court Judge Durwood Pye, reports, irregularities, primary, Mayor-nominate Ivan Allen Jr.	(7.3)

7.1.3 Benefits

On the indexing layer (Figure 7.1), entities and concepts are treated as two specially made sets of *keywords* to process: entities are treated as long nominal compositions in groups, while concepts are considered as ontology tags attached to words.

This approach has three advantages.

1. Processing details are isolated between layers

This method is inspired by the TCP/IP network protocol [Stevens, 1994]. Similar to it, this model ensures that the details on one layer are only internally visible. For example, if the indexing algorithms or internal data structures are changed on the indexing layer, this will not affect the entity and concept extraction as long as the

interfaces and corresponding data structures between these layers are not changed.

2. Reuse of indexing methods

The keyword-based indexing techniques are used three times: in entity indexing, concept indexing and the indexing following their merging. This approach takes advantage of the object-oriented programming paradigm [Eckel, 2000], where software components are reused in different applications to reduce the development time and cost. This makes this knowledge management system compact, easier to update and maintain, and easier to adopt new indexing methods with little redevelopment cost.

3. Ease of adding ontology knowledge into the system

Due to the separation of the tagging and indexing layers, it is not necessary to change any part of the model if additional ontologically tagged text is included. This only requires processing the text according to the procedures described in the next section.

7.2 PROCESSING

The processing is shown in Figure 7.2 and the detailed descriptions of the different information types used in the figure are shown in Table 7.1.

The indexing process involves into two tasks. The first task is to create the stemmed compression dictionary (.text.dict) and store the text into a compression format (.text) to save storage space and more importantly, to improve the retrieval efficiency. The second task is to build the inverted files (.invf) , the document weight file (.weight), and inverted file index (.invf.idx), and to store them, so that the documents could be

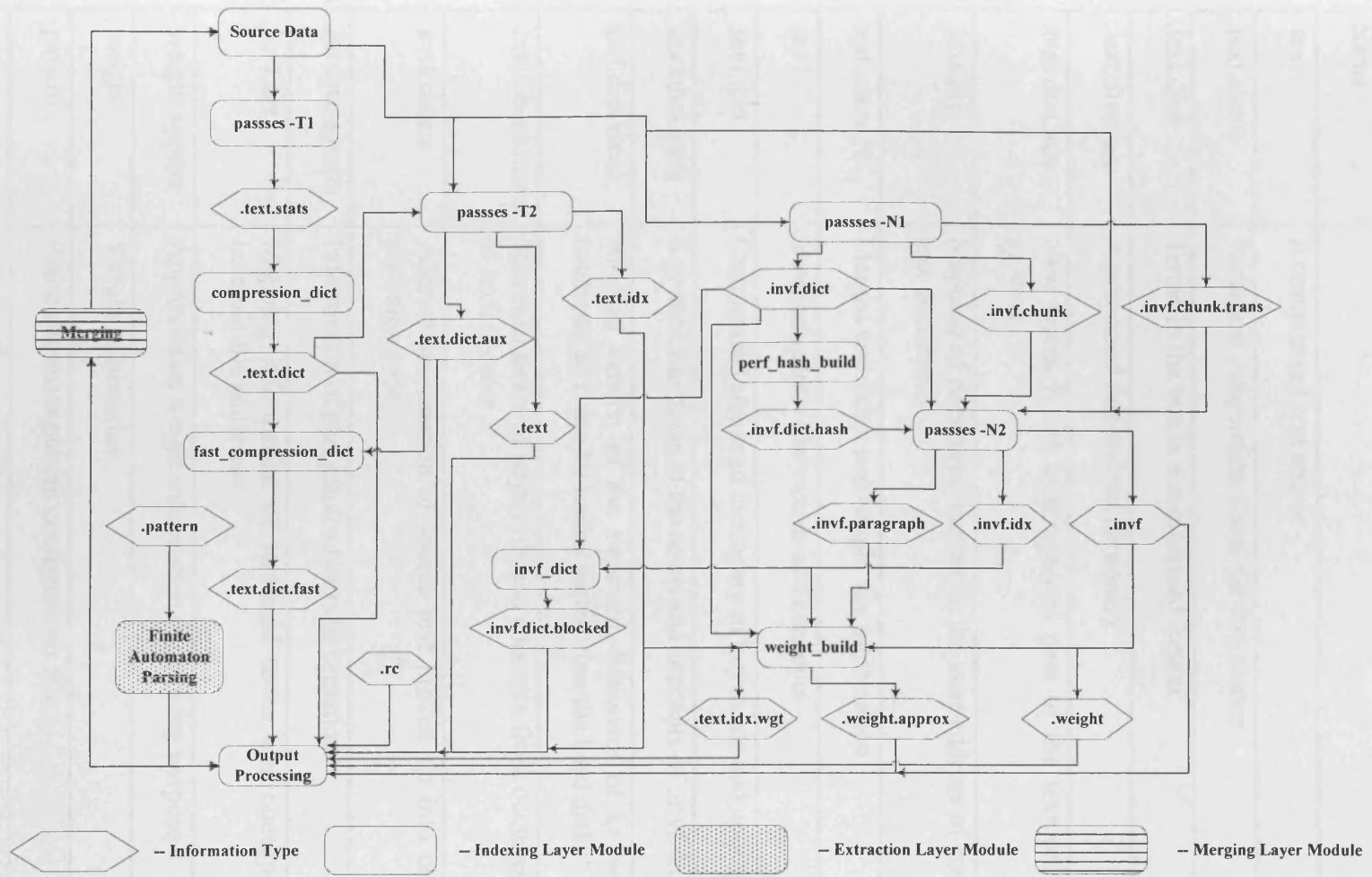


Figure 7.2. Processing

Table 7.1 Information Types Used

Information Name	Description
.text	A compressed text source
.text.status	Statistical information about the text source
.text.dict	Terms in the text in a compressed format
.text.dict.fast	A processed fast-to-load dictionary
.text.dict.aux	New words found in the second pass of the text compression process
.text.idx	Mapping of document number to the start address of compressed text documents
text.idx.wgt	Merged text index and weight file information
.invf	Inverted index of keywords and concepts
.invf.dict	Compressed stemmed dictionary of keywords and concepts
.invf.dict.hash	Keyword functions of the terms and concepts of .invf.dict
invf.dict.block	Blocked version of the stemmed dictionary of keywords and concepts, so it can be loaded faster from the hard disk
.invf.chunk.trans	Stemmed terms of keywords and concepts from occurrence order to lexical order
.invf.chunk	Address information of source text broken up into chunks for inversion pass
.invf.paragraph	Information of paragraph address information
.invf.idx	Mapping information of stemmed terms and concepts to the inverted file addresses
.weight.approx	Approximate weight information for ranking purpose
.weight	Weight information
.pattern	Finite automata pattern configuration file
.rc	Initialisation parameters

retrieved when needed. These two tasks are performed in two steps. In the first step, the subsystem indexes and compresses the text (.text), and in the second step it generates the inverted file (.invf) and the inverted file dictionary (.invf.dict).

7.3 IMPLEMENTATION AND TESTING

The knowledge indexing and retrieving system is built based on the information retrieval system reported by Witten et al. (1999). The system is implemented in C, C Shell and Java on Linux Mandrake 10. Sample source code developed is included in Appendix F.

Tests are conducted on Pentium III M 700 MHz PC with 384MB RAM. Processing speed and computer resource consumption are two main factors measured in the tests. The tests of the indexing speed are divided into 3 parts to measure the speed of entity indexing, concept indexing, and the indexing after merging of the entity and concept indices. The testing data sets used are untagged Brown Corpus (UBC) with approximately one million words. Each data set contains several copies of the UBC. The tests start with the smallest data set, which contains 6 UBC copies (6 million words), and then the size increases by one million words as an increment. The biggest data set used contains 60 copies (60 million words).

7.3.1 Indexing Tests

First, part of speech tagging is conducted on the UBC so that part of speech tags are attached to each word. Ontology tagging is also conducted separately on the UBC to assign ontology tags to each word/phrase in it. Next, indexing of the entities and

concepts are conducted based on these two tagged UBCs. Finally, the information extracted from the two UBCs is merged.

The speed of entity indexing, concept indexing and the indexing after the merging of the former two is illustrated in Figure 7.3, 7.4 and 7.5 respectively. As shown in the figures, the total time for processing over 60 million words using the three procedures is less than 20 minutes.

As shown in those figures, the processing time in the testing sets is approximately proportional to the size of the text data set. It can be expected that the same indexing architecture can be applied to larger text data sets because the processing time only increases proportionally to the size of the text data rather than increasing dramatically.

7.3.2 Testing of Entity and Concept Extraction

The purpose of the second set of tests is to evaluate the speed of retrieving all entities or all concepts in the indexed text collection. Entities are extracted from the UBC which is assigned part of speech tags. Then, the longest nominal word compositions are extracted from each document as entities with a reference to the document itself and output to a file. Next, similarly, all concept tags in each document are extracted with a reference to the document itself, and output to a file.

In these tests, the processing time and computer resource consumptions (such as CPU time, memory usage and disk usage) are monitored and recorded. Figures 7.6 - 7.9 show the testing results for entity extraction, while Figures 7.10 - 7.13 show the corresponding results for concept extraction. These figures, based on the results included in Appendix G, show that in entity and concept extraction, the extraction

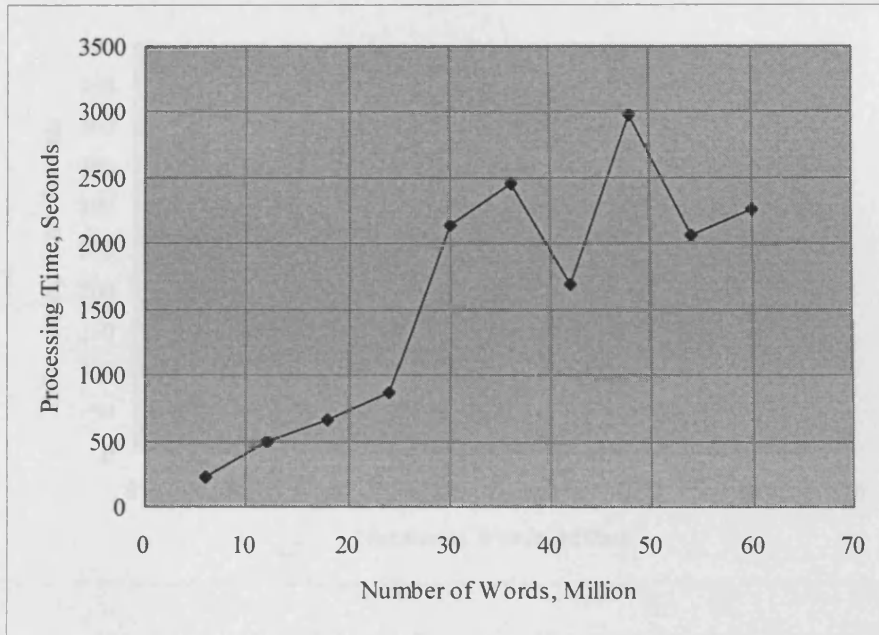


Figure 7.3 Total Processing Time for Entity Indexing

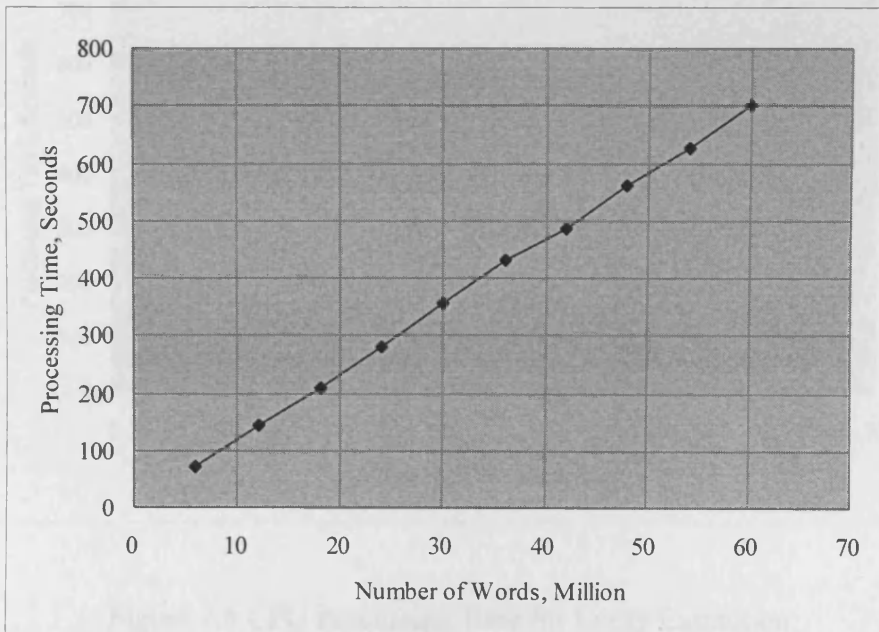


Figure 7.4 Total Processing Time for Concept Indexing

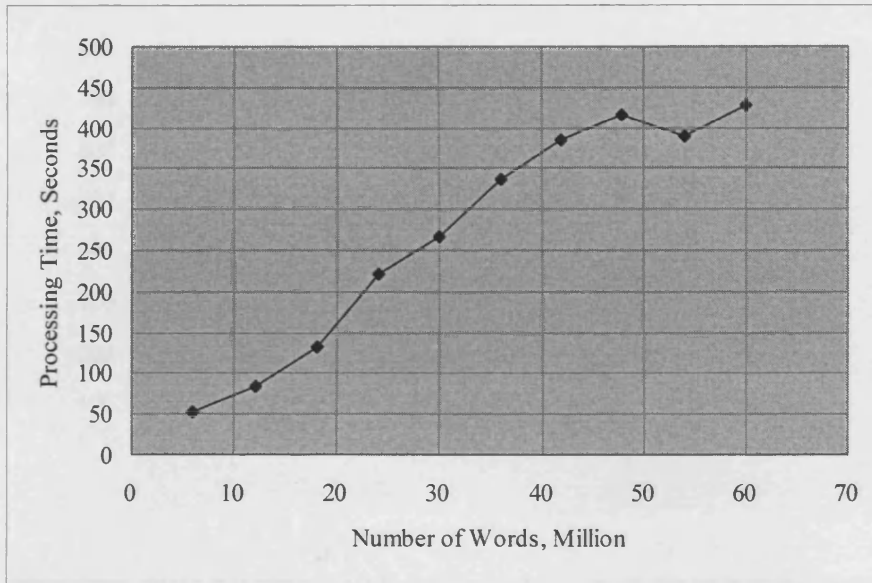


Figure 7.5 Total Processing Time for Merged Indexing

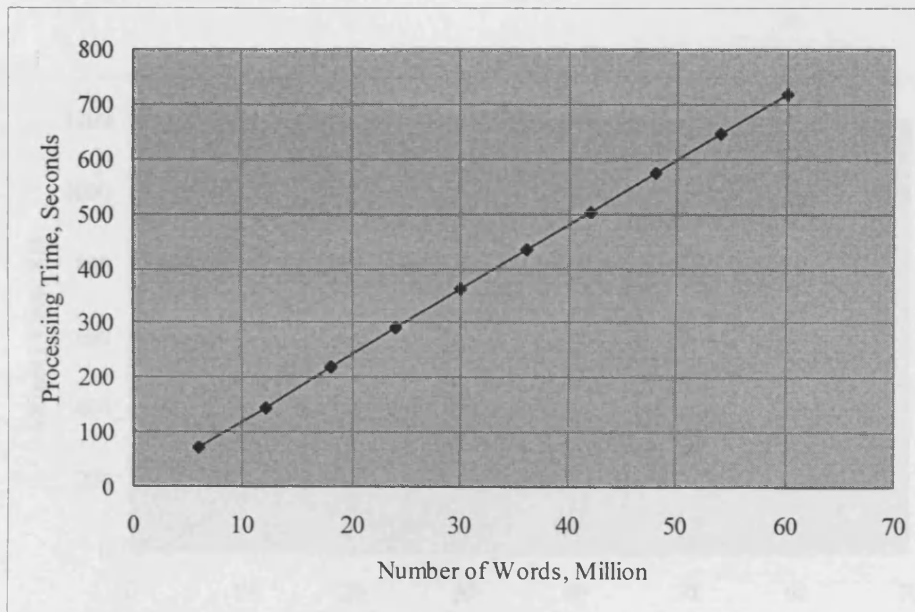


Figure 7.6 CPU Processing Time for Entity Extraction

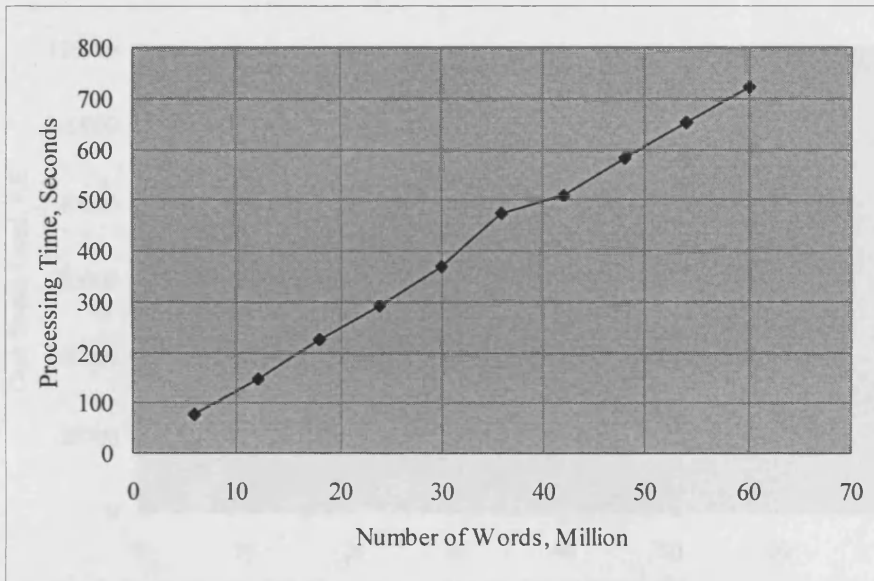


Figure 7.7 Total Processing Time for Entity Extraction

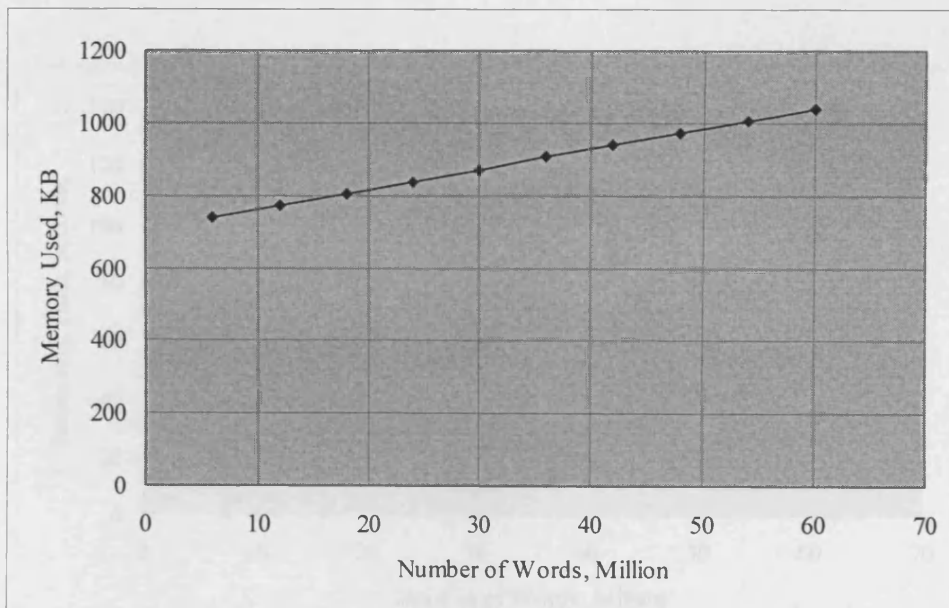


Figure 7.8 Memory Usage for Entity Extraction

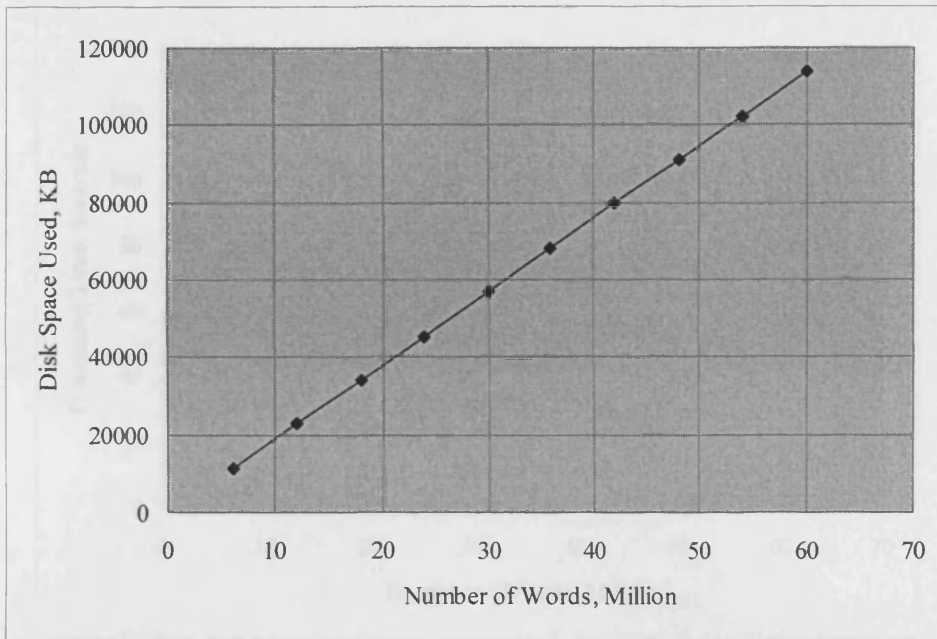


Figure 7.9 Disk Usage for Entity Extraction

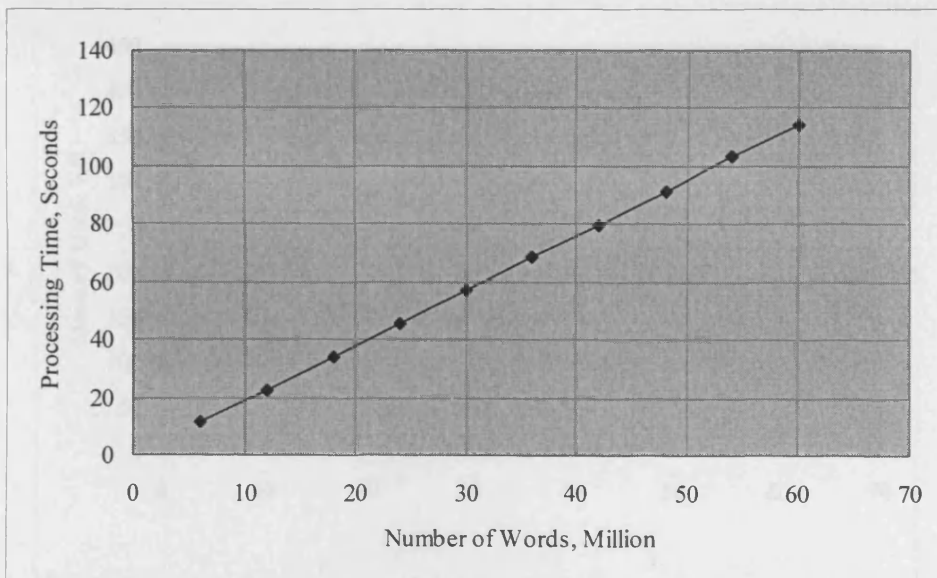


Figure 7.10 CPU Processing Time for Concept Extraction

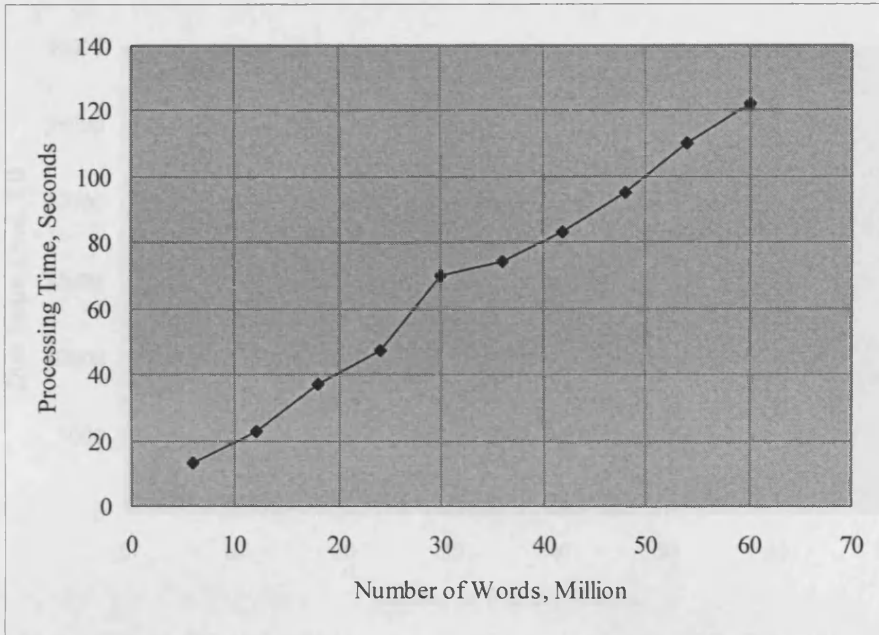


Figure 7.11 Total Processing Time for Concept Extraction

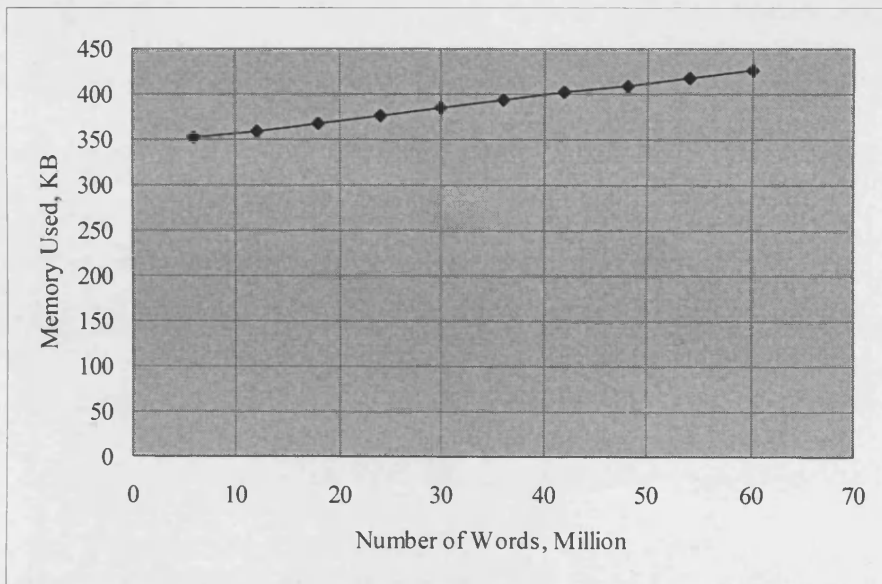


Figure 7.12 Memory Usage for Concept Extraction

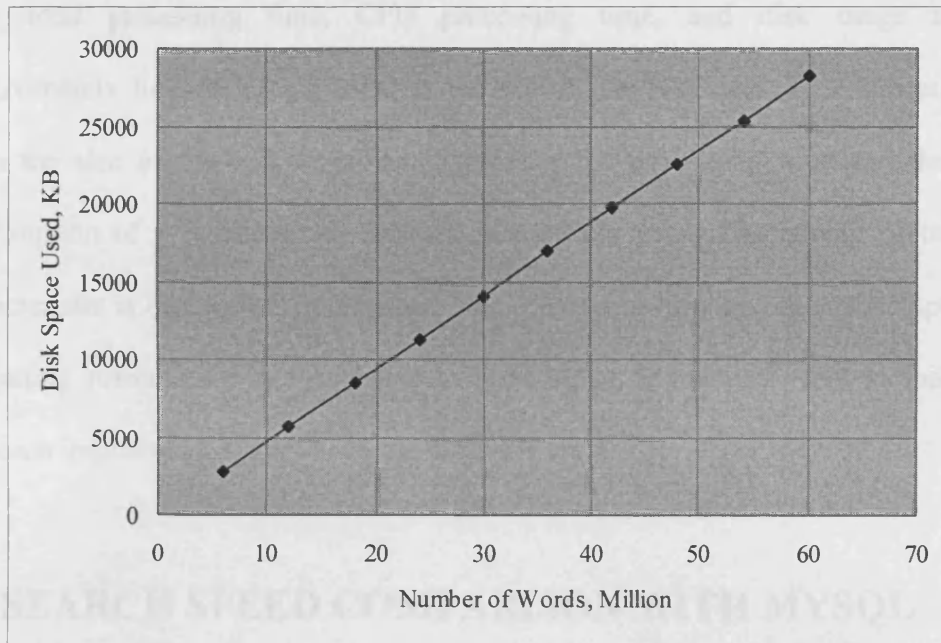


Figure 7.13 Disk Usage for Concept Extraction

time, total processing time, CPU processing time, and disk usage are all approximately linearly proportional to the size of the text data. This indicates that when the size of the text to process increases, the processing time and the other consumption of resources all increase in a steady pace. The benefit of this key characteristic is that in real applications, the processing time and the consumption of computing resources is not expected to increase so dramatically that to make the approach impractical.

7.4 SEARCH SPEED COMPARISON WITH MYSQL

Search speed evaluation is important for a knowledge management system, therefore a benchmark baseline system needs to be determined to compare the test results. MySQL4.0 has been selected for this task as one of the industrial standardised database solutions with full text search function. Therefore, speed tests are conducted in comparison with MySQL4.0 to show the capability of the system developed in this chapter.

Before conducting the tests, the length of a word/concept query needs to be determined. An assumption is made here that the query pattern is expected to follow a typical keyword search query which is 2 to 3 words in average [eTesting Labs Inc, 2000]. Therefore, the length of every query is three words.

Furthermore, there are four considerations related to MySQL which affect the tests to be conducted.

1. Due to the limitation of the MySQL full text search, any word less than 3 letters will not be indexed. Therefore, the word/concept query needs to be longer than 3

letters.

2. Another limitation is related to the use of Boolean queries in MySQL, when sometimes no result is returned, which makes the comparison of query time impossible. Therefore, only relevance ranking method is used in these tests. Similar to the relevance ranking used in MySQL, approximate vector space cosine ranking is used in the system developed in this chapter.

3. Next, as MySQL is running on Windows platform and the system in this chapter is implemented under Linux, irrelevant factors which may affect test results, such as the use of different operating systems should be excluded as much as possible. The query time is recorded when a query is issued until all results are retrieved but have not been output to display. As a result, the differences in the file structure and input/output subsystems in these two different operating systems are eliminated.

4. Indexing of table columns in MySQL is useful as it improves the speed of retrieval. Therefore, the table columns are indexed before the speed tests are conducted.

In the test text collection used, there are over one million words/concepts with different frequencies of appearance. These words are first ordered according to this frequency, and then they are divided into 10 sections. For each section, 15 words/concepts are randomly selected and divided into 5 groups with 3 words/concepts in each of them. The 3 words/concepts in each group are then sent out to the systems as a query. Finally, the average retrieving time is obtained for these 5 groups as shown in Figure 7.14. Tests are conducted using MySQL 4.0.24 on Windows 2000 Professional with the same configuration throughout the tests.

As shown in Figure 7.14, the query time ranges from 20 to 4053.5 milliseconds.

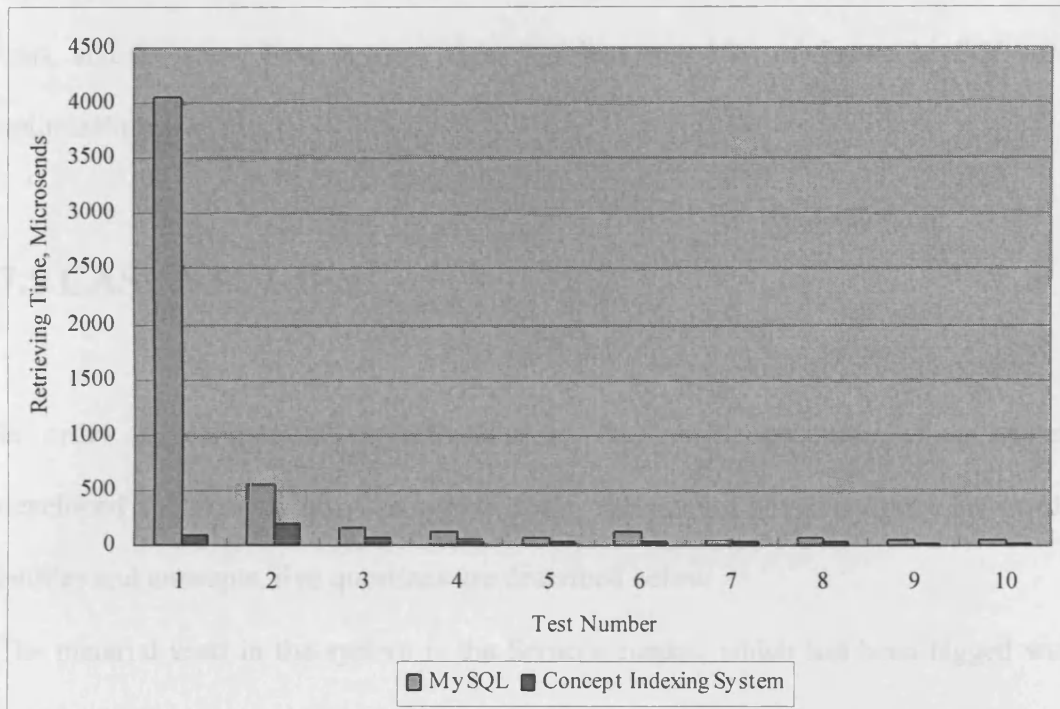


Figure 7.14 Retrieving Time Comparison with MySQL Using Text after Merging of Entity and Concept Indices

Queries with frequently used words tend to need longer time to complete (Test 1 in Figure 7.14). The system developed in this chapter outperformed MySQL in all query tests, and the query time in most cases was less than 50% of that of MySQL after optimisation.

7.5 CASE STUDIES

In order to evaluate the effectiveness of the knowledge management system developed and to show how the system deals with queries which combine keywords, entities and concepts, five questions are described below.

The material used in the system is the Semcor corpus, which has been tagged with POS tags and ontology tags. There are 20,138 lines of senses in Semcor, where each 5 sentences are grouped as one document. Therefore, there are 4,028 documents which contain approximately 440,000 words. The senses in each document are tagged with POS tags, which indicate their parts of speech, and ontology tags (Figure 7.15).

As shown in the figure, each document has three parts, the first part of the document is 5 sentences tagged with part of speech tags; the second part of the document is the same 5 sentences tagged with ontology tags; the last part of the document is the document number. Words with little useful information are tagged with "IGNORE" in a document. The POS tags used follow the format used in Brown Corpus [Francis, and Kucera, 1979]. For example, "Fulton/NP" indicates the word "Fulton" is a proper noun. The ontology tags used are defined in the ontology dictionary OntoRo developed in this thesis. In the case studies, the documents are tagged with ontology tags at Head group level from OntoRo (Figure 5.1). For example, "produced/S306"

The/IGNORE Fulton/NP County/NP Grand/NP Jury/NP said/VB Friday/NN an/IGNORE investigation/NN of/IGNORE Atlanta/NN 's/
IGNORE recent/JJ primary/NN election/NN produced/VB ` /IGNORE no/IGNORE evidence/NN ' /IGNORE that/IGNORE any/IGNORE
irregularities/NN took/VB place/VB . /IGNORE
The/IGNORE jury/NN further/RB said/VB in/IGNORE term/NN end/NN presentments/NN that/IGNORE the/IGNORE City/NP
Executive/NP Committee/NP /IGNORE which/IGNORE had/VB over-all/JJ charge/NN of/IGNORE the/IGNORE election/NN /
IGNORE ' /IGNORE deserves/VB the/IGNORE praise/NN and/IGNORE thanks/NN of/IGNORE the/IGNORE City/NP of/NP Atlanta/NP
/IGNORE for/IGNORE the/IGNORE manner/NN in/IGNORE which/IGNORE the/IGNORE election/NN was/IGNORE conducted/VB /
IGNORE
...

The/IGNORE Fulton/IGNORE County/IGNORE Grand/IGNORE Jury/IGNORE said/S334 Friday/S579 an/IGNORE investigation/S267
of/IGNORE Atlanta/IGNORE 's/IGNORE recent/S74 primary/S362 election/S362 produced/S306 ` /IGNORE no/IGNORE evidence/
S276 ' /IGNORE that/IGNORE any/IGNORE irregularities/S48 took/UNKNOWN place/UNKNOWN . /IGNORE
The/IGNORE jury/S272 further/UNKNOWN said/S334 in/IGNORE term/S65 end/S38 presentments/S561 that/IGNORE the/IGNORE
City/IGNORE Executive/IGNORE Committee/IGNORE . /IGNORE which/IGNORE had/S471 over-all/S45 charge/S455 of/IGNORE
the/IGNORE election/S444 , /IGNORE ' /IGNORE deserves/S562 the/IGNORE praise/S566 and/IGNORE thanks/S557 of/IGNORE
the/IGNORE City/IGNORE of/IGNORE Atlanta/IGNORE ' /IGNORE for/IGNORE the/IGNORE manner/S4 in/IGNORE which/IGNORE
the/IGNORE election/S444 was/IGNORE conducted/S407 . /IGNORE
...

DOCNUM000001

Figure 7.15 The Composition of a Document Used in the Case Study

indicates that word “produced” is related to head group 306, which has the meaning of “disclosure”, “discovery” and “revelation”.

Five test tasks are used to examine the performance of the system and a comparison result is shown in Table 7.2.

Task 1: What aspects are important to choose a good place for water related leisure entertainment?

Find documents which explain what aspects make a place suitable for water related leisure activities should be considered relevant.

Query steps:

1. Use keyword “water”, as it must be in the document. There were **132** documents retrieved.
2. Within the **132** documents obtained in step 1, use “**S513**”, which is the concept number related to the concept of “entertainment, leisure and amusement”, there were **16** documents left.
3. In the **16** documents obtained in step 2, use “**S507**” (interest)¹, there were **4** documents left.
4. In the **4** document left in step 3, use “**S91**” (contribute to, determine, reason), there were **2** documents left as shown in Figure 7.16.

Explanation:

One keyword “water” and three concepts “S513”, “S507”, “S91” were used as queries to restrict documents obtained from 132 to 2 documents. The query can be interpreted as “find documents which contain “water” as an “entertainment” (S513) with some “contributing” factors (S91) to people’s “interest” (S507).

¹ This means “S507” is the concept number related to the concept of “interest”. The rest of the case studies follow the same style.

Table 7.2 Boolean Keyword Query Result

Task Number	The Keywords the Retrieved Documents Must Contain	Number of Documents Retrieved	Relevance (Related/all Retrieved)
1	water, aspect	0	N/A
1	water, activity	0	N/A
1	water, entertainment	0	N/A
1	water, boat	0	N/A
1	water, leisure	0	N/A
1	water, relaxation	0	N/A
1	water, reason	1	0/1
1	water, contribute	2	0/2
1	good, water, sports	0	N/A
1	water, because	4	1/4
2	bizarre, phenomena	0	N/A
2	strange, phenomena	0	N/A
2	strange, sea, lives	0	N/A
2	bizarre, sea, lives	0	N/A
2	weird, sea, lives	0	N/A
2	interesting, phenomena	0	N/A
2	interesting, sea	0	N/A
2	strange, sea, situation	0	N/A
2	strange, sea, case	0	N/A
2	strange, sea anomaly	0	N/A
3	money, genuine	0	N/A
3	money, useless	0	N/A
3	money, not, useful	0	N/A
3	money, profit	3	0/3
3	money, fortune	0	N/A
3	money, fake, product	0	N/A
3	money, fake, service	0	N/A
3	fake, product	0	N/A
3	fake, service	0	N/A
3	make, profit	2	0/2
4	buses	0	N/A
4	cars, economical	0	N/A
4	cars, economic	0	N/A
4	cars, alternative	0	N/A
4	cars, affect	0	N/A
4	transport	3	1/3
4	cars, expensive	0	N/A
4	trucks, expensive	0	N/A
4	trucks, inexpensive	0	N/A
4	trucks, cost	0	N/A
5	religion, modern	3	1/3
5	religion, belief	3	0/3
5	modern, christianity	3	1/3

Table 7.2 Boolean “and” Keyword Query Result (Cont.)

5	religion, change	0	N/A
5	religion, affect	0	N/A
5	Christianity, change	0	N/A
5	Christianity, opinion	0	N/A
5	religion, people, opinion	0	N/A
5	religion, people, think	0	N/A
5	religion, people, different	1	0/1

Result Output
<p>A/IGNORE site/S110 which/IGNORE overlooks/S259 a/IGNORE harbor/IGNORE or/IGNORE river/S204 may/IGNORE offer/S382 interest/S507 in/IGNORE the/IGNORE activities/S372 of/IGNORE boating/S513 traffic/S157 ./IGNORE</p> <p>An/IGNORE area/S110 on/IGNORE the/IGNORE coast/S199 may/IGNORE have/S32 relaxing/S410 views/S255 of/IGNORE the/IGNORE surf/S204 rolling/S152 in/IGNORE on/IGNORE a/IGNORE beach/S19 ./IGNORE</p> <p>A/IGNORE site/S110 may/IGNORE also/S22 be/S335 attractive/S515 just/UNKNOWN through/IGNORE the/IGNORE beauty/S507 of/IGNORE its/IGNORE trees/S214 and/IGNORE shrubs/S214 ./IGNORE</p> <p>Note/S260 extent/S19 of/IGNORE these/IGNORE interests/S507 and/IGNORE how/IGNORE available/S386 they/IGNORE will/IGNORE be/S335 for/IGNORE the/IGNORE public/S30 to/IGNORE enjoy/S506 ./IGNORE</p> <p>Water/S193 interest/S507 is/S1 one/S52 of/IGNORE the/IGNORE most/UNKNOWN valuable/S385 factors/S91 you/IGNORE can/IGNORE find/S92 for/IGNORE a/IGNORE recreation/S513 site/S110 ./IGNORE</p> <p>A/IGNORE site/NN which/IGNORE overlooks/VB a/IGNORE harbor/NN or/IGNORE river/NN may/IGNORE offer/VB interest/NN in/IGNORE the/IGNORE activi/NN ties/NN of/IGNORE boating/NN traffic/NN ./IGNORE</p> <p>An/IGNORE area/NN on/IGNORE the/IGNORE coast/NN may/IGNORE have/VB relaxing/JJ views/NN of/IGNORE the/IGNORE surf/NN rolling/VB in/IGNORE on/IGNORE a/IGNORE beach/NN ./IGNORE</p> <p>A/IGNORE site/NN may/IGNORE also/RB be/VB attractive/JJ just/RB through/IGNORE the/IGNORE beauty/NN of/IGNORE its/IGNORE trees/NN and/IGNORE shrubs/NN ./IGNORE</p> <p>Note/VB extent/NN of/IGNORE these/IGNORE interests/NN and/IGNORE how/IGNORE available/JJ they/IGNORE will/IGNORE be/VB for/IGNORE the/IGNORE public/NN to/IGNORE enjoy/VB ./IGNORE</p> <p>Water/NN interest/NN is/VB one/JJ of/IGNORE the/IGNORE most/RB valuable/JJ factors/NN you/IGNORE can/IGNORE find/VB for/IGNORE a/IGNORE recreation/NN site/NN ./IGNORE</p> <p>DOCNUM00371</p>

Figure 7.16 Result Output and Corresponding Original Documents for Task 1

Determine/S282 how/IGNORE much/IGNORE topography/S136 limits/S452 useful/S386 area/S110 or/IGNORE what/IGNORE the/IGNORE costs/S22 of/IGNORE earth/S158 moving/S38 or/IGNORE grading/S15 might/IGNORE be/IGNORE ./IGNORE

-/IGNORE In/IGNORE addition/IGNORE to/IGNORE its/IGNORE recreation/S513 interests/S507 ./IGNORE water/S25 is/IGNORE needed/S376 for/IGNORE drinking/S174 ./IGNORE sanitation/S392 ./IGNORE and/IGNORE irrigation/S216 ./IGNORE

The/IGNORE quantity/S15 and/IGNORE quality/S3 of/IGNORE water/S25 sources/S91 is/S335 often/UNKNOWN a/IGNORE big/S385 factor/S91 in/IGNORE site/S110 selection/S362 ./IGNORE

The/IGNORE area/S110 may/IGNORE provide/S378 good/S571 springs/S173 or/IGNORE opportunities/S92 for/IGNORE a/IGNORE well/S150 or/IGNORE be/S112 near/S117 to/IGNORE municipal/S108 water/S271 lines/S271 ./IGNORE

Figure/S295 the/IGNORE cost/S22 of/IGNORE providing/S378 water/S25 to/IGNORE the/IGNORE use/S386 areas/S110 ./IGNORE

Determine/VB how/IGNORE much/IGNORE topography/NN limits/VB useful/JJ area/NN or/IGNORE what/IGNORE the/IGNORE costs/NN of/IGNORE earth/NN moving/VB or/IGNORE grading/NN might/IGNORE be/IGNORE ./IGNORE

-/IGNORE In/IGNORE addition/IGNORE to/IGNORE its/IGNORE recreation/NN interests/NN ./IGNORE water/NN is/IGNORE needed/VB for/IGNORE drinking/NN ./IGNORE sanitation/NN ./IGNORE and/IGNORE irrigation/NN ./IGNORE

The/IGNORE quantity/NN and/IGNORE quality/NN of/IGNORE water/NN sources/NN is/VB often/RB a/IGNORE big/JJ factor/NN in/IGNORE site/NN selection/NN ./IGNORE

The/IGNORE area/NN may/IGNORE provide/VB good/JJ springs/NN or/IGNORE opportunities/NN for/IGNORE a/IGNORE well/NN or/IGNORE be/VB near/JJ to/IGNORE municipal/JJ water/NN lines/NN ./IGNORE

Figure/VB the/IGNORE cost/NN of/IGNORE providing/VB water/NN to/IGNORE the/IGNORE use/NN areas/NN ./IGNORE

DOCNUM00381

Figure 7.16 Result Output and Corresponding Original Documents for Task 1 (Cont.)

Original Documents

A site which overlooks a harbor or river may offer interest in the activities of boating traffic .

An area on the coast may have relaxing views of the surf rolling in on a beach .

A site may also be attractive just through the beauty of its trees and shrubs .

Note extent of these interests and how available they will be for the public to enjoy .

Water interest is one of the most valuable factors you can find for a recreation site .

DOCNUM00371

Determine how much topography limits useful area or what the costs of earth moving or grading might be .

- In addition to its recreation interests , water is needed for drinking , sanitation , and irrigation .

The quantity and quality of water sources is often a big factor in site selection .

The area may provide good springs or opportunities for a well or be near to municipal water lines .

Figure the cost of providing water to the use areas .

DOCNUM00381

Figure 7.16 Result Output and Corresponding Original Documents for Task 1 (Cont.)

Task 2: Are there any documents about bizarre phenomena about sea lives? What are the reasons that caused these phenomena?

Find documents which contain the information of strange situations happened in sea or other water bodies where animals lives in, with some explanations of causes of the phenomena.

Query steps:

1. “S204” (sea, river, water etc.) is used as the concept for query, **136** documents were returned.
2. In **136** documents returned, use “S90” (phenomenon) as the query, **43** documents were returned.
3. Use “S91” (cause, reason) as the query, there were **11** documents returned.
4. Further, use “S210” (life, animal, plant) as the query, **2** documents were left.
5. User “S48” (strange, bizarre) as the query, **one** document was left (Figure 7.17).

Explanation:

Five concepts “S204”, “S90”, “S91”, “S210” and “S48” were used to restrict retrieval results from 136 documents to 2 documents. The query can be interpreted as “find documents which describe phenomena (“S90”) happened in sea (“S204”) about animals (“S210”) living in it, and also with some explanation about the cause (“S91”).

Task 3: Why some useless goods or services can make money? Are there any particular people names are mentioned in the documents?

Find any documents which explain why some useless or fake goods or services can make profit.

Result Output

Scientists/S287 and/IGNORE fishermen/S370 have/IGNORE occasionally/UNKNOWN seen/S282 strange/S48 by-products/UNKNOWN of/IGNORE the/IGNORE phenomenon/S90 ./IGNORE During/IGNORE a/IGNORE 1933/IGNORE tsunami/S204 in/IGNORE Japan/S203 the/IGNORE sea/S19 glowed/S222 brilliantly/S222 at/IGNORE night/S243 ./IGNORE The/IGNORE luminosity/S243 of/IGNORE the/IGNORE water/S193 is/IGNORE now/S72 believed/S265 to/IGNORE have/IGNORE been/IGNORE caused/S91 by/IGNORE the/IGNORE stimulation/S101 of/IGNORE vast/S19 numbers/S49 of/IGNORE the/IGNORE luminescent/S243 organism/S210 Noctiluca/S210 miliaris/S210 by/IGNORE the/IGNORE turbulence/S505 of/IGNORE the/IGNORE sea/S19 ./IGNORE

Japanese/S19 fishermen/S370 have/IGNORE sometimes/UNKNOWN observed/S282 that/IGNORE sardines/S175 hauled/S168 up/IGNORE in/IGNORE their/IGNORE nets/S370 during/IGNORE a/IGNORE tsunami/S204 have/S471 enormously/S19 swollen/S116 stomachs/S114 ;/IGNORE the/IGNORE fish/S214 have/IGNORE swallowed/S174 vast/S19 numbers/S49 of/IGNORE bottom/S124 living/S113 diatoms/S113 ,/IGNORE raised/S180 to/IGNORE the/IGNORE surface/S130 by/IGNORE the/IGNORE disturbance/S35 ./IGNORE

The/IGNORE waves/S122 of/IGNORE a/IGNORE 1923/IGNORE tsunami/S204 in/IGNORE Sagami/IGNORE Bay/IGNORE brought/S22 to/IGNORE the/IGNORE surface/S130 and/IGNORE battered/S388 to/IGNORE death/S65 huge/S19 numbers/S49 of/IGNORE fishes/S214 that/IGNORE normally/IGNORE live/S113 at/IGNORE a/IGNORE depth/S130 of/IGNORE 3000/IGNORE feet/S119 ./IGNORE

Scientists/NN and/IGNORE fishermen/NN have/IGNORE occasionally/RB seen/VB strange/JJ by-products/NN of/IGNORE the/IGNORE phenomenon/NN ./IGNORE During/IGNORE a/IGNORE 1933/IGNORE tsunami/NN in/IGNORE Japan/NN the/IGNORE sea/NN glowed/VB brilliantly/RB at/IGNORE night/NN ./IGNORE

The/IGNORE luminosity/NN of/IGNORE the/IGNORE water/NN is/IGNORE now/RB believed/VB to/IGNORE have/IGNORE been/IGNORE caused/VB by/IGNORE the/IGNORE stimulation/NN of/IGNORE vast/JJ numbers/NN of/IGNORE the/IGNORE luminescent/JJ organism/NN Noctiluca/NN miliaris/NN by/IGNORE the/IGNORE turbulence/NN of/IGNORE the/IGNORE sea/NN ./IGNORE

Japanese/JJ fishermen/NN have/IGNORE sometimes/RB observed/VB that/IGNORE sardines/NN hauled/VB up/IGNORE in/IGNORE their/IGNORE nets/NN during/IGNORE a/IGNORE tsunami/NN have/VB enormously/RB swollen/JJ stomachs/NN ;/IGNORE the/IGNORE fish/NN have/IGNORE swallowed/VB vast/JJ numbers/NN of/IGNORE bottom/NN living/JJ diatoms/NN ,/IGNORE raised/VB to/IGNORE the/IGNORE surface/NN by/IGNORE the/IGNORE disturbance/NN ./IGNORE

The/IGNORE waves/NN of/IGNORE a/IGNORE 1923/IGNORE tsunami/NN in/IGNORE Sagami/NP Bay/NP brought/VB to/IGNORE the/IGNORE surface/NN and/IGNORE battered/VB to/IGNORE death/NN huge/JJ numbers/NN of/IGNORE fishes/NN that/IGNORE normally/RB live/VB at/IGNORE a/IGNORE depth/NN of/IGNORE 3000/IGNORE feet/NN ./IGNORE

DOCNUM02551

Figure 7.17 Result Output and Corresponding Original Documents for Task 2

Original Documents

Scientists and fishermen have occasionally seen strange by-products of the phenomenon .

During a 1933 tsunami in Japan the sea glowed brilliantly at night .

The luminosity of the water is now believed to have been caused by the stimulation of vast numbers of the luminescent organism *Noctiluca miliaris* by the turbulence of the sea .

Japanese fishermen have sometimes observed that sardines hauled up in their nets during a tsunami have enormously swollen stomachs ; the fish have swallowed vast numbers of bottom living diatoms , raised to the surface by the disturbance .

The waves of a 1923 tsunami in Sagami Bay brought to the surface and battered to death huge numbers of fishes that normally live at a depth of 3000 feet .

DOCNUM02551

Figure 7.17 Result Output and Corresponding Original Documents for Task 2 (Cont.)

Query steps:

1. Send keyword “money” as a query, **92** documents were returned.
2. In **92** documents returned, send “**S91**” (cause or reason) as a query, **16** documents were left.
3. In **16** documents returned, send “**S317**” (fake, not genuine, and counterfeit) as a query, **3** documents were left (Figure 7.18).
4. From the three documents returned, send a regular expression ² “[A-Z][^]*V(NP)[]?([A-Z][^]*V(NP)[]?)+” (pattern for recognizing name entities) as a query, results are shown in Figure 7.19.

Explanation:

One keyword “money”, two concepts “S91” and “S317” were used to return 3 documents, and one pattern is to extract name entities from the 3 documents retrieved. The query can be interpreted as “find documents which explain why (“S91”) fakes (“S317”) can make “money”. In the documents retrieved, find any people names which follow the pattern beginning with an upper case letter and having a “proper noun” part of speech tag (“NP”).

Task 4: How commuting on road is compared with other commuting methods for a business?

Find documents which contain information about the cost comparison between traveling by road and other traveling methods and how it affects any business involved.

² A concise description of a pattern to be matched during a string search. The pattern is written according to one of several small formal languages devised for this purpose, and may call for matching fixed strings, runs of arbitrary

Result Output

But/IGNORE it/IGNORE is/IGNORE our/IGNORE health/S220 -/IGNORE more/S15 precious/S544
than/IGNORE all/IGNORE the/IGNORE money/S489 in/IGNORE the/IGNORE world/S187 -
/IGNORE that/IGNORE these/IGNORE modern/S72 witch/UNKNOWN doctors/UNKNOWN
with/IGNORE their/IGNORE fake/S317 therapeutic/S395 gadgets/S377 are/IGNORE gambling/S369
away/UNKNOWN ./IGNORE

By/IGNORE preying/S21 on/IGNORE the/IGNORE sick/S391 ./IGNORE by/IGNORE playing/S32
callously/S32 on/IGNORE the/IGNORE hopes/S526 of/IGNORE the/IGNORE
desperate/S46 ./IGNORE by/IGNORE causing/S91 the/IGNORE sufferer/S391 to/IGNORE delay/S74
proper/S387 medical/UNKNOWN care/UNKNOWN ./IGNORE these/IGNORE medical/S395
ghouls/S30 create/S95 pain/S507 and/IGNORE misery/S506 by/IGNORE their/IGNORE very/S19
activity/S372 ./IGNORE

Typically/S372 ./IGNORE Sarah/IGNORE Gross/IGNORE and/IGNORE Mr./IGNORE A/IGNORE
both/IGNORE lost/S301 more/S15 than/IGNORE their/IGNORE money/S489 as/IGNORE
the/IGNORE result/S90 of/IGNORE their/IGNORE experiences/S90 with/IGNORE their/IGNORE
Cleveland/IGNORE quacks/S395 ./IGNORE

Sarah/IGNORE Gross/IGNORE found/S282 that/IGNORE the/IGNORE treatments/S395 given/S461
her/IGNORE for/IGNORE a/IGNORE nervous/S505 ailment/S391 by/IGNORE the/IGNORE
masseur/S194 were/IGNORE not/UNKNOWN helping/S388 her/IGNORE ./IGNORE

As/IGNORE a/IGNORE result/S90 ./IGNORE she/IGNORE consulted/S346 medical/S395
authorities/S395 and/IGNORE learned/S302 that/IGNORE the/IGNORE devices/S377 her/IGNORE
quack/S286 ./IGNORE doctor/S395 ./IGNORE was/IGNORE using/S405 were/S335
phony/S12 ./IGNORE

But/IGNORE it/IGNORE is/IGNORE our/IGNORE health/NN -/IGNORE more/JJ precious/JJ
than/IGNORE all/IGNORE the/IGNORE money/NN in/IGNORE the/IGNORE world/NN -/IGNORE
that/IGNORE these/IGNORE modern/JJ witch/NN doctors/NN with/IGNORE their/IGNORE fake/JJ
therapeutic/JJ gadgets/NN are/IGNORE gambling/VB away/RB ./IGNORE

By/IGNORE preying/VB on/IGNORE the/IGNORE sick/JJ ./IGNORE by/IGNORE playing/VB
callously/RB on/IGNORE the/IGNORE hopes/NN of/IGNORE the/IGNORE desperate/NN ./IGNORE
by/IGNORE causing/VB the/IGNORE sufferer/NN to/IGNORE delay/VB proper/JJ medical/NN
care/NN ./IGNORE these/IGNORE medical/JJ ghouls/NN create/VB pain/NN and/IGNORE
misery/NN by/IGNORE their/IGNORE very/JJ activity/NN ./IGNORE

Typically/RB ./IGNORE Sarah/NP Gross/NP and/IGNORE Mr./NP A/NP both/IGNORE lost/VB
more/NN than/IGNORE their/IGNORE money/NN as/IGNORE the/IGNORE result/NN of/IGNORE
their/IGNORE experiences/NN with/IGNORE their/IGNORE Cleveland/NN quacks/NN ./IGNORE

Sarah/NP Gross/NP found/VB that/IGNORE the/IGNORE treatments/NN given/VB her/IGNORE
for/IGNORE a/IGNORE nervous/JJ ailment/NN by/IGNORE the/IGNORE masseur/NN
were/IGNORE not/RB helping/VB her/IGNORE ./IGNORE

As/IGNORE a/IGNORE result/NN ./IGNORE she/IGNORE consulted/VB medical/JJ authorities/NN
and/IGNORE learned/VB that/IGNORE the/IGNORE devices/NN her/IGNORE quack/JJ ./IGNORE
doctor/NN ./IGNORE was/IGNORE using/VB were/VB phony/JJ ./IGNORE

DOCNUM00451

Figure 7.18 Result Output and Corresponding Original Documents for Task 3, Step 3

Enthusiastically/IGNORE ,/IGNORE Americans/IGNORE have/IGNORE swept/UNKNOWN
 subliterate/UNKNOWN and/IGNORE bogus/S317 materials/S1 like/IGNORE Paul/IGNORE
 Bunyan/IGNORE tales/S351 ,/IGNORE Abe/S351 Lincoln/S351 anecdotes/S351 and/IGNORE
 labor/IGNORE union/IGNORE songs/S240 up/IGNORE as/IGNORE true/S288 products/S95
 of/IGNORE our/IGNORE American/IGNORE oral/S343 tradition/S95 ./IGNORE
 Nor/IGNORE have/IGNORE we/IGNORE remembered/S265 that/IGNORE in/IGNORE the/IGNORE
 melting/UNKNOWN pot/UNKNOWN of/IGNORE America/IGNORE the/IGNORE hundreds/S59
 of/IGNORE isolated/S26 and/IGNORE semi/IGNORE isolated/S26 ethnic/S6 ,/IGNORE regional/S30
 and/IGNORE occupational/S372 groups/IGNORE did/IGNORE not/UNKNOWN fuse/S25
 into/IGNORE a/IGNORE homogeneous/S11 national/S113 unit/S217 until/IGNORE
 long/UNKNOWN after/IGNORE education/S286 and/IGNORE industrialization/S95 had/IGNORE
 caused/S91 them/IGNORE to/IGNORE cast/IGNORE oral/S343 tradition/S95 aside/IGNORE
 as/IGNORE a/IGNORE means/S378 of/IGNORE carrying/S301 culturally/S301 significant/S301
 material/S1 ./IGNORE
 Naturally/UNKNOWN ,/IGNORE such/S44 scholarly/S314 facts/S90 are/S335 of/IGNORE little/S19
 concern/S506 to/IGNORE the/IGNORE man/S100 trying/S403 to/IGNORE make/S470 money/S489
 or/IGNORE fan/S365 patriotism/S570 by/IGNORE means/S378 of/IGNORE folklore/S286 ./IGNORE
 That/IGNORE much/S19 of/IGNORE what/IGNORE he/IGNORE calls/S46 folklore/S286 is/S335
 the/IGNORE result/S90 of/IGNORE beliefs/S276 carefully/S266 sown/S134 among/IGNORE
 the/IGNORE people/S217 with/IGNORE the/IGNORE conscious/S186 aim/S369 of/IGNORE
 producing/S97 a/IGNORE desired/S376 mass/S19 emotional/S103 reaction/S91 to/IGNORE
 a/IGNORE particular/S9 situation/S4 or/IGNORE set/S44 of/IGNORE situations/S4 is/S335
 irrelevant/S5 ./IGNORE
 As/IGNORE long/IGNORE as/IGNORE his/IGNORE material/S1 is/S335 Americana/S335 ,/IGNORE
 can/IGNORE in/S92 some/S92 way/S92 be/IGNORE ascribed/S92 to/IGNORE the/IGNORE
 masses/S19 and/IGNORE appears/S259 ' /IGNORE democratic/S16 ' /IGNORE to/IGNORE
 his/IGNORE audience/S309 ,/IGNORE he/IGNORE remains/S85 satisfied/S529 ./IGNORE
 Enthusiastically/RB ,/IGNORE Americans/NN have/IGNORE swept/VB subliterate/JJ and/IGNORE
 bogus/JJ materials/NN like/IGNORE Paul/NP Bunyan/NP tales/NN ,/IGNORE Abe/NN Lincoln/NN
 anecdotes/NN and/IGNORE labor/NN union/NN songs/NN up/IGNORE as/IGNORE true/JJ
 products/NN of/IGNORE our/IGNORE American/JJ oral/JJ tradition/NN ./IGNORE
 Nor/IGNORE have/IGNORE we/IGNORE remembered/VB that/IGNORE in/IGNORE the/IGNORE
 melting/NN pot/NN of/IGNORE America/NN the/IGNORE hundreds/NN of/IGNORE isolated/JJ
 and/IGNORE semi/JJ isolated/JJ ethnic/JJ ,/IGNORE regional/JJ and/IGNORE occupational/JJ
 groups/NP did/IGNORE not/RB fuse/VB into/IGNORE a/IGNORE homogeneous/JJ national/JJ
 unit/NN until/IGNORE long/RB after/IGNORE education/NN and/IGNORE industrialization/NN
 had/IGNORE caused/VB them/IGNORE to/IGNORE cast/VB oral/JJ tradition/NN aside/IGNORE
 as/IGNORE a/IGNORE means/NN of/IGNORE carrying/VB culturally/RB significant/JJ
 material/NN ./IGNORE

Figure 7.18 Result Output and Corresponding Original Documents for Task 3, Step 3

(Cont.)

Naturally/RB ,/IGNORE such/JJ scholarly/JJ facts/NN are/VB of/IGNORE little/JJ concern/NN to/IGNORE the/IGNORE man/NN trying/VB to/IGNORE make/VB money/NN or/IGNORE fan/VB patriotism/NN by/IGNORE means/NN of/IGNORE folklore/NN ./IGNORE
That/IGNORE much/NN of/IGNORE what/IGNORE he/IGNORE calls/VB folklore/NN is/VB the/IGNORE result/NN of/IGNORE beliefs/NN carefully/RB sown/VB among/IGNORE the/IGNORE people/NN with/IGNORE the/IGNORE conscious/JJ aim/NN of/IGNORE producing/VB a/IGNORE desired/JJ mass/JJ emotional/JJ reaction/NN to/IGNORE a/IGNORE particular/JJ situation/NN or/IGNORE set/NN of/IGNORE situations/NN is/VB irrelevant/JJ ./IGNORE
As/IGNORE long/IGNORE as/IGNORE his/IGNORE material/NN is/VB Americana/NN ,/IGNORE can/IGNORE in/RB some/RB way/RB be/IGNORE ascribed/VB to/IGNORE the/IGNORE masses/NN and/IGNORE appears/VB `/IGNORE democratic/JJ '/IGNORE to/IGNORE his/IGNORE audience/NN ,/IGNORE he/IGNORE remains/VB satisfied/JJ ./IGNORE
DOCNUM00466

The/IGNORE collector/S46 enjoys/S506 the/IGNORE contact/S432 with/IGNORE rural/S216 life/S4 ;/IGNORE he/IGNORE hunts/S370 folklore/S286 for/IGNORE the/IGNORE very/S19 `/IGNORE field/IGNORE and/IGNORE stream/IGNORE '/IGNORE reasons/S91 that/IGNORE many/S19 persons/IGNORE hunt/S370 game/S370 ;/IGNORE and/IGNORE only/UNKNOWN rarely/UNKNOWN is/S335 he/IGNORE acutely/S19 concerned/S19 with/IGNORE the/IGNORE meaning/S301 of/IGNORE what/IGNORE he/IGNORE has/IGNORE located/S282 ./IGNORE
Fundamentally/UNKNOWN ,/IGNORE both/IGNORE these/IGNORE types/S46 ,/IGNORE the/IGNORE amateur/S369 and/IGNORE the/IGNORE collector/S46 ,/IGNORE are/S335 uncritical/S385 and/IGNORE many/IGNORE of/IGNORE them/IGNORE do/IGNORE n't/IGNORE distinguish/S302 well/UNKNOWN between/IGNORE real/S1 folklore/S286 and/IGNORE bogus/S317 material/S1 ./IGNORE
But/IGNORE there/IGNORE are/IGNORE also/S22 the/IGNORE commercial/S372 propagandists/S309 and/IGNORE the/IGNORE analysts/S292 -/IGNORE one/IGNORE dominated/IGNORE by/IGNORE money/S489 ,/IGNORE the/IGNORE other/IGNORE by/IGNORE nineteenth/IGNORE century/S66 German/S6 scholarship/S286 ./IGNORE
Both/IGNORE are/S335 primarily/UNKNOWN concerned/UNKNOWN with/IGNORE the/IGNORE uses/S405 that/IGNORE can/IGNORE be/IGNORE made/S26 of/IGNORE the/IGNORE material/S1 that/IGNORE the/IGNORE collector/S46 has/IGNORE found/S470 ./IGNORE
Both/IGNORE shudder/S185 at/IGNORE the/IGNORE thought/S261 of/IGNORE proceeding/S167 too/UNKNOWN far/UNKNOWN beyond/IGNORE the/IGNORE sewage/IGNORE system/IGNORE and/IGNORE the/IGNORE electric/S93 light/S243 lines/S93 ./IGNORE

Figure 7.18 Result Output and Corresponding Original Documents for Task 3, Step 3

(Cont.)

The/IGNORE collector/NN enjoys/VB the/IGNORE contact/NN with/IGNORE rural/JJ life/NN ;/IGNORE he/IGNORE hunts/VB folklore/NN for/IGNORE the/IGNORE very/JJ `/IGNORE field/IGNORE and/IGNORE stream/IGNORE ' /IGNORE reasons/NN that/IGNORE many/JJ persons/NP hunt/VB game/NN ;/IGNORE and/IGNORE only/RB rarely/RB is/VB he/IGNORE acutely/RB concerned/JJ with/IGNORE the/IGNORE meaning/NN of/IGNORE what/IGNORE he/IGNORE has/IGNORE located/VB ./IGNORE

Fundamentally/RB ,/IGNORE both/IGNORE these/IGNORE types/NN ,/IGNORE the/IGNORE amateur/NN and/IGNORE the/IGNORE collector/NN ,/IGNORE are/VB uncritical/JJ and/IGNORE many/IGNORE of/IGNORE them/IGNORE do/IGNORE n't/RB distinguish/VB well/RB between/IGNORE real/JJ folklore/NN and/IGNORE bogus/JJ material/NN ./IGNORE

But/IGNORE there/IGNORE are/IGNORE also/RB the/IGNORE commercial/JJ propagandists/NN and/IGNORE the/IGNORE analysts/NN -/IGNORE one/IGNORE dominated/JJ by/JJ money/NN ,/IGNORE the/IGNORE other/IGNORE by/IGNORE nineteenth/JJ century/NN German/JJ scholarship/NN ./IGNORE

Both/IGNORE are/VB primarily/RB concerned/JJ with/IGNORE the/IGNORE uses/NN that/IGNORE can/IGNORE be/IGNORE made/VB of/IGNORE the/IGNORE material/NN that/IGNORE the/IGNORE collector/NN has/IGNORE found/VB ./IGNORE

Both/IGNORE shudder/VB at/IGNORE the/IGNORE thought/NN of/IGNORE proceeding/VB too/RB far/RB beyond/IGNORE the/IGNORE sewage/NN system/NN and/IGNORE the/IGNORE electric/JJ light/NN lines/NN ./IGNORE

DOCNUM00470

Figure 7.18 Result Output and Corresponding Original Documents for Task 3, Step 3

(Cont.)

Original Documents

But it is our health - more precious than all the money in the world - that these modern witch doctors with their fake therapeutic gadgets are gambling away .

By preying on the sick , by playing callously on the hopes of the desperate , by causing the sufferer to delay proper medical care , these medical ghouls create pain and misery by their very activity .

Typically , Sarah Gross and Mr. A both lost more than their money as the result of their experiences with their Cleveland quacks .

Sarah Gross found that the treatments given her for a nervous ailment by the masseur were not helping her .

As a result , she consulted medical authorities and learned that the devices her quack ` doctor ' was using were phony .

DOCNUM00451

Enthusiastically , Americans have swept subliterate and bogus materials like Paul Bunyan tales , Abe Lincoln anecdotes and labor union songs up as true products of our American oral tradition .

Nor have we remembered that in the melting pot of America the hundreds of isolated and semi isolated ethnic , regional and occupational groups did not fuse into a homogeneous national unit until long after education and industrialization had caused them to cast oral tradition aside as a means of carrying culturally significant material .

Naturally , such scholarly facts are of little concern to the man trying to make money or fan patriotism by means of folklore .

That much of what he calls folklore is the result of beliefs carefully sown among the people with the conscious aim of producing a desired mass emotional reaction to a particular situation or set of situations is irrelevant .

As long as his material is Americana , can in some way be ascribed to the masses and appears ` democratic ' to his audience , he remains satisfied .

DOCNUM00466

The collector enjoys the contact with rural life ; he hunts folklore for the very ` field and stream ' reasons that many persons hunt game ; and only rarely is he acutely concerned with the meaning of what he has located .

Fundamentally , both these types , the amateur and the collector , are uncritical and many of them do n't distinguish well between real folklore and bogus material .

But there are also the commercial propagandists and the analysts - one dominated by money , the other by nineteenth century German scholarship .

Both are primarily concerned with the uses that can be made of the material that the collector has found .

Both shudder at the thought of proceeding too far beyond the sewage system and the electric light lines .

DOCNUM00470

Figure 7.18 Result Output and Corresponding Original Documents for Task 3, Step 3

(Cont.)

Result Output
<p>Sarah/NP Gross/NP ,Mr./NP A/NP ,Sarah/NP Gross/NP , DOCNUM00451</p> <p>Paul/NP Bunyan/NP , DOCNUM00466</p>
Original Documents
<p>Sarah Gross, Mr. A, Sarah Gross DOCNUM00451</p> <p>Paul Bunyan, DOCNUM00466</p>

Figure 7.19 Result Output and Corresponding Original Documents for Task 3, Step 4

Query steps:

1. Send “S157” (travel and traffic) as a query, **402** documents were returned.
2. From the **402** documents obtained in step 1, send “S372” (business) as a query, **65** documents were returned.
3. From **65** documents retrieved in step 2, send “S498” (cost, expense) as a query, **2** documents were retrieved.
4. From **2** documents obtained from step 3, “S165” (road) was sent as query, **one** document was left as the result (Figure 7.20)

Explanation:

Four concepts “S157”, “S372”, “S498” and “S165” were used in queries to restrict documents from 402 documents to 1 document. The query can be interpreted as “find documents which compare the cost (“S498”) of traveling (“S157”) on the road (“S165”) or other methods for business “S372” purpose.

Task 5: How people think the religion nowadays?

Find documents related to people’s belief, religion, and philosophy, and possibly the reasons that science affects people’s way of thinking.

Query steps:

1. Send “S261” (idealism) as the query, **266** documents were retrieved.
2. From the **266** documents obtained in step 1, send “S91” (cause, reason) as the query, **81** documents were returned.
3. From the **81** documents obtained in step 2, send “S598” (religion) as the query, **11** documents were left.
4. Use the keywords “spirit”, “bible” or “science” as the query, so that the documents returned must contain at least one keyword from these three keywords. There

Result Output

The/IGNORE railroad/IGNORE siding/IGNORE is/S335 still/UNKNOWN important/S385 -/IGNORE it/IGNORE is/S335 usually/S335 ,/IGNORE though/IGNORE not/UNKNOWN always/UNKNOWN ,/IGNORE true/S288 that/IGNORE long-haul/UNKNOWN shipment/S160 by/IGNORE rail/S126 is/S335 cheaper/S498 than/IGNORE trucking/IGNORE ./IGNORE But/IGNORE anybody/IGNORE who/IGNORE promises/S464 a/IGNORE substantial/S19 volume/S19 of/IGNORE business/S372 can/IGNORE get/S91 a/IGNORE railroad/S374 to/IGNORE run/S20 a/IGNORE short/S19 spur/S374 to/IGNORE his/IGNORE plant/S413 these/IGNORE days/S66 ,/IGNORE and/IGNORE many/S19 businesses/S372 can/IGNORE live/S65 without/IGNORE the/IGNORE railroad/S374 ./IGNORE And/IGNORE there/IGNORE are/S90 now/S72 many/S19 millions/S59 of/IGNORE workers/S359 for/IGNORE whom/IGNORE the/IGNORE factory/S95 with/IGNORE the/IGNORE big/S19 parking/UNKNOWN lot/UNKNOWN ,/IGNORE which/IGNORE can/IGNORE be/IGNORE reached/S172 by/IGNORE driving/S157 across/IGNORE or/IGNORE against/IGNORE the/IGNORE usual/S48 pattern/S374 of/IGNORE rush/UNKNOWN hour/UNKNOWN traffic/S157 and/IGNORE grille/S155 route/S165 bus/S165 lines/S165 ,/IGNORE is/S335 actually/UNKNOWN more/UNKNOWN convenient/S220 than/IGNORE the/IGNORE walk-to/IGNORE factory/S95 ./IGNORE Willow/IGNORE Run/IGNORE ,/IGNORE General/IGNORE Electric/IGNORE 's/IGNORE enormous/S19 installations/S477 at/IGNORE Louisville/IGNORE and/IGNORE Syracuse/IGNORE ,/IGNORE the/IGNORE Pentagon/S113 ,/IGNORE Boeing/IGNORE in/IGNORE Seattle/IGNORE ,/IGNORE Douglas/IGNORE and/IGNORE Lockheed/IGNORE in/IGNORE Los/IGNORE Angeles/IGNORE ,/IGNORE the/IGNORE new/S75 automobile/S161 assembly/S95 plants/S413 everywhere/UNKNOWN -/IGNORE none/IGNORE of/IGNORE these/IGNORE is/IGNORE substantially/UNKNOWN served/S386 by/IGNORE any/IGNORE sort/S44 of/IGNORE conventional/S48 mass/S48 rapid/S48 transit/S48 ./IGNORE They/IGNORE are/S335 all/IGNORE suburban/S134 plants/S413 ,/IGNORE relying/UNKNOWN on/UNKNOWN the/IGNORE roads/S165 to/IGNORE keep/S452 them/IGNORE supplied/S378 with/IGNORE workers/S359 ./IGNORE The/IGNORE railroad/NN siding/NN is/VB still/RB important/JJ -/IGNORE it/IGNORE is/VB usually/RB ,/IGNORE though/IGNORE not/RB always/RB ,/IGNORE true/JJ that/IGNORE long-haul/NN shipment/NN by/IGNORE rail/NN is/VB cheaper/JJ than/IGNORE trucking/NN ./IGNORE But/IGNORE anybody/IGNORE who/IGNORE promises/VB a/IGNORE substantial/JJ volume/NN of/IGNORE business/NN can/IGNORE get/VB a/IGNORE railroad/NN to/IGNORE run/VB a/IGNORE short/JJ spur/NN to/IGNORE his/IGNORE plant/NN these/IGNORE days/NN ,/IGNORE and/IGNORE many/JJ businesses/NN can/IGNORE live/VB without/IGNORE the/IGNORE railroad/NN ./IGNORE

Figure 7.20 Result Output and Corresponding Original Documents for Task 4

And/IGNORE there/IGNORE are/VB now/RB many/JJ millions/NN of/IGNORE workers/NN for/IGNORE whom/IGNORE the/IGNORE factory/NN with/IGNORE the/IGNORE big/JJ parking/NN lot/NN ,/IGNORE which/IGNORE can/IGNORE be/IGNORE reached/VB by/IGNORE driving/VB across/IGNORE or/IGNORE against/IGNORE the/IGNORE usual/JJ pattern/NN of/IGNORE rush/NN hour/NN traffic/NN and/IGNORE grille/NN route/NN bus/NN lines/NN ,/IGNORE is/VB actually/RB more/RB convenient/JJ than/IGNORE the/IGNORE walk-to/JJ factory/NN ./IGNORE

Willow/NP Run/NP ,/IGNORE General/NP Electric/NP 's/IGNORE enormous/JJ installations/NN at/IGNORE Louisville/NP and/IGNORE Syracuse/NP ,/IGNORE the/IGNORE Pentagon/NN ,/IGNORE Boeing/NP in/IGNORE Seattle/NP ,/IGNORE Douglas/NP and/IGNORE Lockheed/NP in/IGNORE Los/NN Angeles/NN ,/IGNORE the/IGNORE new/JJ automobile/NN assembly/NN plants/NN everywhere/RB -/IGNORE none/IGNORE of/IGNORE these/IGNORE is/IGNORE substantially/RB served/VB by/IGNORE any/IGNORE sort/NN of/IGNORE conventional/JJ mass/NN rapid/NN transit/NN ./IGNORE

They/IGNORE are/VB all/IGNORE suburban/JJ plants/NN ,/IGNORE relying/VB on/VB the/IGNORE roads/NN to/IGNORE keep/VB them/IGNORE supplied/VB with/IGNORE workers/NN ./IGNORE

DOCNUM001088

Original Documents

The railroad siding is still important - it is usually , though not always , true that long-haul shipment by rail is cheaper than trucking .

But anybody who promises a substantial volume of business can get a railroad to run a short spur to his plant these days , and many businesses can live without the railroad .

And there are now many millions of workers for whom the factory with the big parking lot , which can be reached by driving across or against the usual pattern of rush hour traffic and grille route bus lines , is actually more convenient than the walk-to factory .

Willow Run , General Electric 's enormous installations at Louisville and Syracuse , the Pentagon , Boeing in Seattle , Douglas and Lockheed in Los Angeles , the new automobile assembly plants everywhere - none of these is substantially served by any sort of conventional mass rapid transit .

They are all suburban plants , relying on the roads to keep them supplied with workers .

DOCNUM001088

Figure 7.20 Result Output and Corresponding Original Documents for Task 4 (Cont.)

were 3 documents returned (Figure 7.21).

Explanation:

Three concepts “S261”, “S91”, “S598” were used in the queries, and also there are three optional keyword “spirit”, “bible” and “science” were used in the queries as well. The query can be interpreted as “find documents which are related to religion (“S598”) and idealism (“S261”) and explanation (“S91”), and also at least contains one keyword out of “spirit”, “bible” or “science”.

7.6 DISCUSSIONS

This section discusses the benefits of the system.

The system implemented in this chapter performs automatic entity and concept indexing, entity and concept extraction, and supports entity, concept, keyword queries.

This means that the system can have the advantages of keyword-based systems while

Result Output

In/IGNORE such/S44 a/IGNORE world/S90 the/IGNORE words/S114 `/IGNORE matter/S19
'/IGNORE and/IGNORE `/IGNORE spirit/S301 '/IGNORE both/IGNORE referred/S5 to/IGNORE
directly/UNKNOWN known/S302 realities/S90 in/IGNORE the/IGNORE common/S427
experience/S286 of/IGNORE all/IGNORE ./IGNORE

In/IGNORE it/IGNORE important/S385 elements/S33 of/IGNORE Christianity/S598 and/IGNORE
of/IGNORE the/IGNORE Biblical/S599 view/S372 of/IGNORE reality/S90 in/S90
general/S90 ,/IGNORE which/IGNORE now/S72 cause/S91 us/IGNORE much/UNKNOWN
difficulty/S423 ,/IGNORE could/IGNORE be/IGNORE responded/S267 to/IGNORE
quite/UNKNOWN naturally/UNKNOWN and/IGNORE spontaneously/UNKNOWN ./IGNORE

The/IGNORE progress/S21 of/IGNORE science/S261 over/IGNORE these/IGNORE last/S74 few/S62
centuries/S66 and/IGNORE the/IGNORE gradual/S15 replacement/S88 of/IGNORE Biblical/S599
by/IGNORE scientific/S301 categories/S44 of/IGNORE reality/S90 have/IGNORE to/IGNORE
a/IGNORE large/S19 extent/S19 emptied/S174 the/IGNORE spirit/IGNORE world/IGNORE
of/IGNORE the/IGNORE entities/S1 which/IGNORE previously/S71 populated/S113
it/IGNORE ./IGNORE

In/IGNORE carrying/UNKNOWN out/UNKNOWN this/IGNORE program/S414 science/S261
has/IGNORE undoubtedly/S288 performed/S386 a/IGNORE very/S19 considerable/S19 service/S46
for/IGNORE which/IGNORE it/IGNORE can/IGNORE claim/S474 due/S493 credit/S566 ./IGNORE

The/IGNORE objectification/S11 of/IGNORE the/IGNORE world/S217 of/IGNORE spirit/S301
in/IGNORE popular/S444 superstition/S286 had/IGNORE certainly/UNKNOWN gone/S407
far/UNKNOWN beyond/UNKNOWN what/IGNORE the/IGNORE experience/S286 of/IGNORE
spirit/S301 could/IGNORE justify/S272 or/IGNORE support/S272 ./IGNORE

In/IGNORE such/JJ a/IGNORE world/NN the/IGNORE words/NN `/IGNORE matter/NN '/IGNORE
and/IGNORE `/IGNORE spirit/NN '/IGNORE both/IGNORE referred/VB to/IGNORE directly/RB
known/JJ realities/NN in/IGNORE the/IGNORE common/JJ experience/NN of/IGNORE
all/IGNORE ./IGNORE

In/IGNORE it/IGNORE important/JJ elements/NN of/IGNORE Christianity/NN and/IGNORE
of/IGNORE the/IGNORE Biblical/JJ view/NN of/IGNORE reality/NN in/RB general/RB ,/IGNORE
which/IGNORE now/RB cause/VB us/IGNORE much/RB difficulty/NN ,/IGNORE could/IGNORE
be/IGNORE responded/VB to/IGNORE quite/RB naturally/RB and/IGNORE
spontaneously/RB ./IGNORE

The/IGNORE progress/NN of/IGNORE science/NN over/IGNORE these/IGNORE last/JJ few/JJ
centuries/NN and/IGNORE the/IGNORE gradual/JJ replacement/NN of/IGNORE Biblical/JJ
by/IGNORE scientific/JJ categories/NN of/IGNORE reality/NN have/IGNORE to/IGNORE
a/IGNORE large/JJ extent/NN emptied/VB the/IGNORE spirit/NN world/NN of/IGNORE
the/IGNORE entities/NN which/IGNORE previously/RB populated/VB it/IGNORE ./IGNORE

Figure 7.21 Result Output and Corresponding Original Documents for Task 5

In/IGNORE carrying/VB out/VB this/IGNORE program/NN science/NN has/IGNORE undoubtedly/RB performed/VB a/IGNORE very/RB considerable/JJ service/NN for/IGNORE which/IGNORE it/IGNORE can/IGNORE claim/VB due/JJ credit/NN ./IGNORE
 The/IGNORE objectification/NN of/IGNORE the/IGNORE world/NN of/IGNORE spirit/NN in/IGNORE popular/JJ superstition/NN had/IGNORE certainly/RB gone/VB far/RB beyond/RB what/IGNORE the/IGNORE experience/NN of/IGNORE spirit/NN could/IGNORE justify/VB or/IGNORE support/VB ./IGNORE
 DOCNUM00247

In/IGNORE one/S46 debate/S277 he/IGNORE supported/S568 the/IGNORE freedom/S563 of/IGNORE judgment/S283 as/IGNORE opposed/IGNORE to/IGNORE dogma/S598 ./IGNORE in/IGNORE another/IGNORE he/IGNORE held/S399 that/IGNORE the/IGNORE practice/S374 of/IGNORE science/S261 was/S1 in/UNKNOWN fact/UNKNOWN an/IGNORE act/S100 of/IGNORE religious/S603 worship/S372 ./IGNORE
 He/IGNORE continued/S85 as/IGNORE a/IGNORE popular/S531 lecturer/S343 ./IGNORE
 He/IGNORE devised/S373 a/IGNORE detonating/IGNORE fuse/IGNORE in/IGNORE which/IGNORE a/IGNORE short/S19 wire/S120 was/IGNORE caused/S91 to/IGNORE glow/S222 by/IGNORE an/IGNORE electric/S93 current/S93 ./IGNORE
 In/IGNORE 1819/IGNORE under/IGNORE royal/S449 command/S446 he/IGNORE undertook/S407 a/IGNORE very/S19 successful/S91 geological/S272 expedition/S157 to/IGNORE Bornholm/IGNORE ./IGNORE one/S52 of/IGNORE the/IGNORE Danish/IGNORE islands/S199 ./IGNORE being/S1 one/S52 of/IGNORE three/S52 scientists/S287 in/IGNORE the/IGNORE expedition/S157 ./IGNORE
 It/IGNORE was/IGNORE with/IGNORE the/IGNORE assistance/S424 of/IGNORE one/S52 of/IGNORE the/IGNORE members/S33 of/IGNORE this/IGNORE expedition/S157 ./IGNORE Lauritz/IGNORE Esmarch/IGNORE ./IGNORE that/IGNORE Oersted/IGNORE succeeded/S172 in/IGNORE producing/S95 light/S243 by/IGNORE creating/S95 an/IGNORE electric/IGNORE discharge/IGNORE in/IGNORE mercury/S30 vapor/S30 through/IGNORE which/IGNORE an/IGNORE electric/S93 current/S93 was/IGNORE made/S91 to/IGNORE flow/S156 ./IGNORE
 In/IGNORE one/JJ debate/NN he/IGNORE supported/VB the/IGNORE freedom/NN of/IGNORE judgment/NN as/IGNORE opposed/IGNORE to/IGNORE dogma/NN ./IGNORE in/IGNORE another/IGNORE he/IGNORE held/VB that/IGNORE the/IGNORE practice/NN of/IGNORE science/NN was/VB in/RB fact/RB an/IGNORE act/NN of/IGNORE religious/JJ worship/NN ./IGNORE
 He/IGNORE continued/VB as/IGNORE a/IGNORE popular/JJ lecturer/NN ./IGNORE
 He/IGNORE devised/VB a/IGNORE detonating/NN fuse/NN in/IGNORE which/IGNORE a/IGNORE short/JJ wire/NN was/IGNORE caused/VB to/IGNORE glow/VB by/IGNORE an/IGNORE electric/NN current/NN ./IGNORE
 In/IGNORE 1819/IGNORE under/IGNORE royal/JJ command/NN he/IGNORE undertook/VB a/IGNORE very/RB successful/JJ geological/JJ expedition/NN to/IGNORE Bornholm/NP ./IGNORE one/JJ of/IGNORE the/IGNORE Danish/JJ islands/NN ./IGNORE being/VB one/JJ of/IGNORE three/JJ scientists/NN in/IGNORE the/IGNORE expedition/NN ./IGNORE

Figure 7.21 Result Output and Corresponding Original Documents for Task 5 (Cont.)

It/IGNORE was/IGNORE with/IGNORE the/IGNORE assistance/NN of/IGNORE one/JJ of/IGNORE the/IGNORE members/NN of/IGNORE this/IGNORE expedition/NN ,/IGNORE Lauritz/NP Esmarch/NP ,/IGNORE that/IGNORE Oersted/NP succeeded/VB in/IGNORE producing/VB light/NN by/IGNORE creating/VB an/IGNORE electric/NN discharge/NN in/IGNORE mercury/NN vapor/NN through/IGNORE which/IGNORE an/IGNORE electric/NN current/NN was/IGNORE made/VB to/IGNORE flow/VB ./IGNORE
 DOCNUM02300

Certainly/UNKNOWN one/IGNORE of/IGNORE the/IGNORE most/UNKNOWN important/S385 comments/S305 that/IGNORE can/IGNORE be/IGNORE made/IGNORE upon/IGNORE the/IGNORE spiritual/S598 and/IGNORE cultural/S541 life/S4 of/IGNORE any/IGNORE period/S65 of/IGNORE Western/S351 civilization/S217 during/IGNORE the/IGNORE past/S74 sixteen/S74 or/IGNORE seventeen/IGNORE centuries/S66 has/S378 to/IGNORE do/S302 with/IGNORE the/IGNORE way/S4 in/IGNORE which/IGNORE its/IGNORE leaders/IGNORE have/IGNORE read/S305 and/IGNORE interpreted/S305 the/IGNORE Bible/S599 ./IGNORE

This/IGNORE reading/S286 and/IGNORE the/IGNORE comments/S305 that/IGNORE it/IGNORE evoked/S504 constitute/S32 the/IGNORE influence/S91 ./IGNORE

A/IGNORE contrast/S269 of/IGNORE the/IGNORE scripture/S599 reading/S286 of/IGNORE ,/IGNORE let/IGNORE us/IGNORE say/IGNORE ,/IGNORE St./IGNORE Augustine/IGNORE ,/IGNORE John/IGNORE Bunyan/IGNORE ,/IGNORE and/IGNORE Thomas/IGNORE Jefferson/IGNORE ,/IGNORE all/IGNORE three/S55 of/IGNORE whom/IGNORE found/S282 in/IGNORE such/S44 study/S261 a/IGNORE real/S12 source/S91 of/IGNORE enlightenment/S286 ,/IGNORE can/IGNORE tell/S308 us/IGNORE a/IGNORE great/S19 deal/S19 about/IGNORE these/IGNORE three/S52 men/S79 and/IGNORE the/IGNORE age/S66 that/IGNORE each/IGNORE represented/S322 and/IGNORE helped/S386 bring/S90 to/IGNORE conscious/S219 expression/S312 ./IGNORE

In/IGNORE much/UNKNOWN the/IGNORE same/S16 way/S4 ,/IGNORE we/IGNORE recognize/S286 the/IGNORE importance/S531 of/IGNORE Shakespeare/IGNORE 's/IGNORE familiarity/S286 with/IGNORE Plutarch/IGNORE and/IGNORE Montaigne/IGNORE ,/IGNORE of/IGNORE Shelley/IGNORE 's/IGNORE study/S351 of/IGNORE Plato/IGNORE 's/IGNORE dialogues/S346 ,/IGNORE and/IGNORE of/IGNORE Coleridge/IGNORE 's/IGNORE enthusiastic/S504 plundering/IGNORE of/IGNORE the/IGNORE writings/S350 of/IGNORE many/S19 philosophers/S287 and/IGNORE theologians/S598 from/IGNORE Plato/IGNORE to/IGNORE Schelling/IGNORE and/IGNORE William/IGNORE Godwin/IGNORE ,/IGNORE through/IGNORE which/IGNORE so/UNKNOWN many/S19 abstract/S325 ideas/S261 were/IGNORE brought/S22 to/IGNORE the/IGNORE attention/S265 of/IGNORE English/S305 men/UNKNOWN of/UNKNOWN letters/UNKNOWN ./IGNORE

We/IGNORE may/IGNORE also/S22 recognize/S286 cases/S48 in/IGNORE which/IGNORE the/IGNORE poets/S354 have/IGNORE influenced/S100 the/IGNORE philosophers/S287 and/IGNORE even/UNKNOWN indirectly/UNKNOWN the/IGNORE scientists/S287 ./IGNORE

Figure 7.21 Result Output and Corresponding Original Documents for Task 5 (Cont.)

Certainly/RB one/IGNORE of/IGNORE the/IGNORE most/RB important/JJ comments/NN that/IGNORE can/IGNORE be/IGNORE made/IGNORE upon/IGNORE the/IGNORE spiritual/JJ and/IGNORE cultural/JJ life/NN of/IGNORE any/IGNORE period/NN of/IGNORE Western/JJ civilization/NN during/IGNORE the/IGNORE past/JJ sixteen/JJ or/IGNORE seventeen/JJ centuries/NN has/VB to/IGNORE do/VB with/IGNORE the/IGNORE way/NN in/IGNORE which/IGNORE its/IGNORE leaders/NN have/IGNORE read/VB and/IGNORE interpreted/VB the/IGNORE Bible/NN ./IGNORE

This/IGNORE reading/NN and/IGNORE the/IGNORE comments/NN that/IGNORE it/IGNORE evoked/VB constitute/VB the/IGNORE influence/NN ./IGNORE

A/IGNORE contrast/NN of/IGNORE the/IGNORE scripture/NN reading/NN of/IGNORE ,/IGNORE let/IGNORE us/IGNORE say/IGNORE ,/IGNORE St./NP Augustine/NP ,/IGNORE John/NP Bunyan/NP ,/IGNORE and/IGNORE Thomas/NP Jefferson/NP ,/IGNORE all/IGNORE three/NN of/IGNORE whom/IGNORE found/VB in/IGNORE such/JJ study/NN a/IGNORE real/JJ source/NN of/IGNORE enlightenment/NN ,/IGNORE can/IGNORE tell/VB us/IGNORE a/IGNORE great/NN deal/NN about/IGNORE these/IGNORE three/JJ men/NN and/IGNORE the/IGNORE age/NN that/IGNORE each/IGNORE represented/VB and/IGNORE helped/VB bring/VB to/IGNORE conscious/JJ expression/NN ./IGNORE

In/IGNORE much/RB the/IGNORE same/JJ way/NN ,/IGNORE we/IGNORE recognize/VB the/IGNORE importance/NN of/IGNORE Shakespeare/NN 's/IGNORE familiarity/NN with/IGNORE Plutarch/NP and/IGNORE Montaigne/NP ,/IGNORE of/IGNORE Shelley/NP 's/IGNORE study/NN of/IGNORE Plato/NN 's/IGNORE dialogues/NN ,/IGNORE and/IGNORE of/IGNORE Coleridge/NN 's/IGNORE enthusiastic/JJ plundering/IGNORE of/IGNORE the/IGNORE writings/NN of/IGNORE many/JJ philosophers/NN and/IGNORE theologians/NN from/IGNORE Plato/NN to/IGNORE Schelling/NP and/IGNORE William/NP Godwin/NP ,/IGNORE through/IGNORE which/IGNORE so/RB many/JJ abstract/JJ ideas/NN were/IGNORE brought/VB to/IGNORE the/IGNORE attention/NN of/IGNORE English/JJ men/NN of/NN letters/NN ./IGNORE

We/IGNORE may/IGNORE also/RB recognize/VB cases/NN in/IGNORE which/IGNORE the/IGNORE poets/NN have/IGNORE influenced/VB the/IGNORE philosophers/NN and/IGNORE even/RB indirectly/RB the/IGNORE scientists/NN ./IGNORE

DOCNUM02943

Figure 7.21 Result Output and Corresponding Original Documents for Task 5 (Cont.)

Original Documents

In such a world the words 'matter' and 'spirit' both referred to directly known realities in the common experience of all .

In its important elements of Christianity and of the Biblical view of reality in general , which now cause us much difficulty , could be responded to quite naturally and spontaneously .

The progress of science over these last few centuries and the gradual replacement of Biblical by scientific categories of reality have to a large extent emptied the spirit world of the entities which previously populated it .

In carrying out this program science has undoubtedly performed a very considerable service for which it can claim due credit .

The objectification of the world of spirit in popular superstition had certainly gone far beyond what the experience of spirit could justify or support .

DOCNUM00247

In one debate he supported the freedom of judgment as opposed to dogma , in another he held that the practice of science was in fact an act of religious worship .

He continued as a popular lecturer .

He devised a detonating fuse in which a short wire was caused to glow by an electric current .

In 1819 under royal command he undertook a very successful geological expedition to Bornholm , one of the Danish islands , being one of three scientists in the expedition .

It was with the assistance of one of the members of this expedition , Lauritz Esmarch , that Oersted succeeded in producing light by creating an electric discharge in mercury vapor through which an electric current was made to flow .

DOCNUM02300

Certainly one of the most important comments that can be made upon the spiritual and cultural life of any period of Western civilization during the past sixteen or seventeen centuries has to do with the way in which its leaders have read and interpreted the Bible .

This reading and the comments that it evoked constitute the influence .

A contrast of the scripture reading of , let us say , St. Augustine , John Bunyan , and Thomas Jefferson , all three of whom found in such study a real source of enlightenment , can tell us a great deal about these three men and the age that each represented and helped bring to conscious expression .

In much the same way , we recognize the importance of Shakespeare 's familiarity with Plutarch and Montaigne , of Shelley 's study of Plato 's dialogues , and of Coleridge 's enthusiastic plundering of the writings of many philosophers and theologians from Plato to Schelling and William Godwin , through which so many abstract ideas were brought to the attention of English men of letters .

We may also recognize cases in which the poets have influenced the philosophers and even indirectly the scientists .

DOCNUM02943

Figure 7.21 Result Output and Corresponding Original Documents for Task 5 (Cont.)

making use of entities and concepts when the user has query needs that are not easy to describe with keywords only. Examples include finding abstract concepts or discovering relations.

As shown in the Table 7.2 and the five tasks conducted, sometimes keywords can not be used effectively when queries are “vague” and can not be expressly easily only using keywords. In this case, queries which contain both keywords and concepts can help finding documents which “conceptually” correspond to the queries raised. For instance, in the first question, “What aspects are important to choose a good place for water sports (water related leisure activities)?”, the only keyword which can be easily pointed out is “water”, while the concepts of “aspects”, “good”, “factors”, “water sports”, “leisure activities” can not be easily used to generate relevant keywords for querying the system. There are many cases where users ask “how”, “for what reason”, “why” questions rather than simple “what” questions. Even when users want to ask “what” questions, sometimes they do not know the “exact” terms which should be used. A common case is that when people want to know a specific term for a certain thing. For example, it will be difficult for people who want to find out the terminology of “at an equal rate or pace”, which implies fairness and impartiality, whereby creditors are repaid proportionally according to their original investment, which is “*pari passu*” (a term used in banking). In these cases, combined queries with keywords, entities and concepts could be the solution as shown in 5 query tasks described above.

Indexing techniques are used in this system to make it faster, but at the same time, this also provides ways to make the system more flexible. In the entity and concept extraction process, once the index is constructed, there is no need to rebuild it again if different types of entity or concept patterns are used to extract different entity or

concept knowledge. This gives certain advantage in terms of system maintenance and redeployment as it eliminates the need for reprocessing previously processed information.

This system extracts the knowledge and codifies the knowledge into entities and concepts. This is very closed to the view of Kakabadse, who stated that knowledge can be represented as objectively defined and codified concepts and facts [Kakabadse et al., 2003]. The system's simple structure of entities and concepts enables it to organise the codified knowledge in a more dynamic way than previous methods of static and hierarchical concept indexing methods such as [Holub, 2003].

7.7 SUMMARY

In this chapter, a conceptual model and a system architecture for entity and concept-based knowledge management are proposed and implemented. Tests on speed and computer resource consumption are conducted. A comparison with an industrial database solution is made to show the speed improvement. The advantages of using this architecture for knowledge management are also discussed.

CHAPTER 8. CONTRIBUTIONS, CONCLUSIONS AND FUTURE WORK

This chapter summarises the contributions made and conclusions reached and suggests possible directions for further research.

8.1 CONTRIBUTIONS

The main contribution of this research is the development of techniques of ontology, natural language processing and machine learning for knowledge management. These techniques have enabled the automation of knowledge acquisition from unstructured Web documents and facilitated knowledge extraction, organisation and reuse. The specific contributions are summarised below.

1. Formal representation of text-based content through entities and concepts

A conceptual model for knowledge representation of text-based content is developed which describes the content using concepts and entities. In the context of this thesis, an entity is an identifiable and discrete instance existing in a text document, and a concept is the abstract or generalised idea about entities and the relations between them. The conceptual model is further developed into a mathematic model, which formally introduces the assumptions and constrains used and formalises the way knowledge is represented. The use of entities and concepts and the mathematical model developed facilitate ontology tagging, knowledge extraction, and knowledge reuse.

2. Method for automatic knowledge acquisition based on intelligent focused crawling

This method employs a supervised machine learning algorithm, the Support Vector Machine, to the document finding process, so that only relevant documents are retrieved and analysed. This enables the automation of the knowledge acquisition process for large and unstructured text-based content collections such as those on the Web.

3. General purpose ontology dictionary

This ontology dictionary is based on a systematically studied ontological structure. The dictionary has several features including ontological structure with fixed depth, unification of words/phrases that have the same semantic meaning regardless of their different parts of speech, and the use of implicit semantic relations. This dictionary facilitates the measuring of the semantic similarity between words/phrases, and provides new opportunities for introducing more and much richer semantic relations than those currently in use.

4. Method for ontology mapping between a lexicon and an ontology dictionary

This method uses a heuristic measure to find the one-to-one mapping between the entries in the lexicon and the ontology dictionary. The method generates the mapping without any supervised training or the use of other lexicon resources. The mapping together with a lexicon and a semantic concordance corpus provide means to produce an ontologically tagged corpus for automatic ontology tagging.

5. Algorithm for automatic text ontology tagging

The algorithm is a supervised machine learning algorithm which assigns ontology tags to words/phrases in a free text collection. First, training is performed to obtain two types of information from the ontologically tagged training set, i.e. statistical and word/phrase mutual information. Then the algorithm uses the information obtained to automatically assign ontology tags to the text. The algorithm enables the automation of knowledge extraction using the ontology information produced in the tagging process.

6. Method for knowledge extraction and organisation

This method includes three phases: extraction, indexing, and merging. First, entities and concepts are extracted; they are then indexed; and finally, in the merging phase, the entity and concept indices are merged. This method enables efficient knowledge extraction, and the organisation of a large knowledge base where knowledge is represented using entities and concepts.

8.2 CONCLUSIONS

1. There is a growing trend moving from information-centred society towards knowledge-centred society. Four technologies (information retrieval, machine learning, natural language processing, the Semantic Web and the ontology) are now moving closer to support and complement each other to facilitate process of transforming data into knowledge in the chain of knowledge flow.

2. The provision of knowledge management needs four technologies (information retrieval, machine learning, natural language processing, the Semantic Web and the ontology) to be integrated in a systematical way to facilitate the knowledge flow.
3. Conventional knowledge representation models have deficiencies for knowledge management and a new knowledge representation model to facilitate knowledge management is needed.
4. The model of concept indexing meets the requirements of knowledge management.
5. The use of machine learning and focused crawling enables automatic knowledge acquisition.
6. Ontology tagging enables text to be searched, browsed and analysed at different abstraction levels, so that existing text-based document collections can be reused in knowledge management applications.
7. An ontology dictionary and an ontologically tagged corpus are two resources enabling automatic ontology tagging.
8. A rule-based supervised machine learning algorithm can be exploited for ontology tagging with an ontology dictionary and an ontologically tagged corpus developed.
9. The system using concept indexing can provide effective knowledge extraction, organisation and management.

8.3 IMPLEMENTATION WORK

The implementation work conducted in this research is summarised as below.

1. An HTML to plain text converter in Flex language

The purpose of this converter is to remove HTML tags and their values from the Web documents retrieved. This converter is used in several places in the prototype system where the conversion from HTML Web pages to plain text is needed. This converter is used in A and C in Figure 1.1.

2. A focused Web crawling system with graphical user interface implemented in Java

This crawling system enables simultaneous Web document retrieval with the speed of 1000 Web pages per hour in test environment. Various configurations could be set in the graphical user interface with message logging function. This system is located in B in Figure 1.1.

3. A support vector machine training data converter implemented in Java

In this implementation, Web document are converted into a special format required by support vector machine, a supervised machine learning algorithm for training and document classification purpose. The implementation has 5 versions which correspond to 5 options used in two case studies in Chapter 4. This converter is used in C for building training examples in C in Figure 1.1.

4. A modified morphology analyser implemented in Flex¹ language

This implementation is modified based on the morphology analyser in [Minnen et al., 2001], where the original one conducts morphology analysis. In the modified implementation, morphology analyser can not only analyse both words with part of speech tags, but also can analyse words with ontology tags. Input and output formats have been changed to conform to the format of Semcor corpus. Also, nominal possessive pronouns and possessive pronouns are now morphologised to their standard forms. This analyser is used for conducting morphological analysis in C and E.

5. OntoRo, an ontology dictionary

This dictionary contains 228,571 word and phrase entries which are extracted from Roget's Thesaurus [Roget, 2003], a well known thesaurus to serve the purpose in this research work. This dictionary is located in F in Figure 1.1.

6. eWord, a machine readable dictionary

This dictionary contains 77,022 entries extracted from WordNet [Fellbaum, 1998], a widely used lexical database to serve the purpose in this research work. This dictionary is used in G in Figure 1.1.

7. A WordNet entry extractor implemented in Java language

¹ Flex is a fast lexical analyser generator language which is compiled and translated to C to generate the lexical analyser.

The purpose of this implementation is to extract WordNet entries from its specially formed database, so that eWord could be built. This extractor is used in G in Figure 1.1 for building eWord.

8. OntoCorp, an ontologically tagged corpus

This corpus contains 20,138 sentences which are created automatically from a widely used part of speech tagging corpus to serve the purpose in this research work. This corpus is used in G in Figure 1.1.

9. A semantic mapping algorithm implemented in C language

This algorithm implementation maps every entry in eWord to OntoRo for the research conducted in this thesis. This algorithm is used in G in Figure 1.1.

10. A Sencor entry converter implemented in Java language

The purpose of this implementation is to convert the specially formatted Sencor tagged entries to ontologically tagged corpus for training purpose. This converter makes use of the eWord, OntoRo and Sencor as input to generate the OntoCorp. This implementation is used in building ontologically tagged corpus shown in G in Figure 1.1.

11. An ontology tagging algorithm implemented in Java language

This ontology tagging algorithm uses eWord, OntoRo and OntoCorp to learn ontology tagging rules, and uses these rules to tag unseen text. The algorithm implementation has three different versions which correspond to three case studies conducted in Chapter 6. This algorithm is used in E in Figure 1.1.

12. A modified part of speech tagger

In this modified version of Brill's part of speech tagger [Brill, 1992], input and output formats have been changed to conform the description format of Semcor [Fellbaum, 1998], a semantically tagged corpus. Obsolete functions are discarded and the whole package is recompiled. This tagger is used in D in Figure 1.1.

13. A knowledge management system capable of concept indexing and entity, concept and keyword combined search implemented in C language

This system is built based on a demo system used in [Witten et al., 1999], where entity indexing, concept indexing, entity extraction, concept extraction capabilities are added. In addition, the combined query using entities, concepts and keywords are implemented and added to the system so that queries can contain entities, concepts and keywords at the same time. Besides all the function improvements, obsolete functions are removed to improve the system compatibility with the current Linux operating systems. This system is used in H, I and L in Figure 1.1.

8.4 FUTURE WORK

As the current prototype system developed in this work contains many modules, which are command line based, an integrated graphic user interface could be constructed to integrate the interactions between different modules which are previously linked by manually typing commands and running batch command scripts, so that the whole knowledge management process in this system could be streamlined and more accessible.

An open sourced architecture based on the prototype system developed in this thesis could be designed, so that more researchers could benefit from this system and a community of both academia and industry could contribute to the improvement of this system.

The current focused crawling module was evaluated through relatively small scale case studies, therefore larger scale case studies are needed to evaluate its effectiveness in a more realistic environment for practical applications.

Currently, the crawling module only retrieves text document from the Web, in the future it could be extended for crawling other materials from the Web, such as multimedia content.

The ontologically tagged corpus for ontology tagging purpose is automatically converted using a heuristic approach, therefore the corpus is not completely accurate, which could introduce errors when this corpus is used by the ontology tagging algorithm developed to tag unseen text or Web documents. Due to this reason, another task in the future is to calibrate this automatically generated training corpus by using

manual, semi-automatic or full automatic method to correct tagging errors. After the calibration, the tagging benchmarking tests need to be conducted again to investigate whether the tagging accuracy after the correction of the training corpus has improved. Currently, the ontology tagging method produces tagging accuracy approximately 78%, there is space left to improve the tagging accuracy by using other information in the ontologically tagged corpus, or use the information in different ways. Moreover, other machine learning algorithms could be used to improve the tagging accuracy. The ontology dictionary and ontology tagging developed in this work could be made available as web services so that those users who want the ontology definitions and ontology tagging functions could integrate those functionalities seamlessly into their applications without deploying the same modules again.

Furthermore, a possible research opportunity is to investigate the applicability of this system into other engineering areas, such as conceptual design and technical trouble shooting, where vast technology knowledge needs to be searched and exploited in an efficient way.

Another research direction is the applications related to the Semantic Web, which has received increased attention. For example, there is an ample space to be explored for effectively transforming the existing Web content smoothly into the content on the Semantic Web by transforming the ontologically tagged text content to RDF or OWL based content, which is compatible with the Semantic Web specifications.

APPENDIX A. MATERIALS AND RESULTS

A.1 POSITIVE AND NEGATIVE EXAMPLES

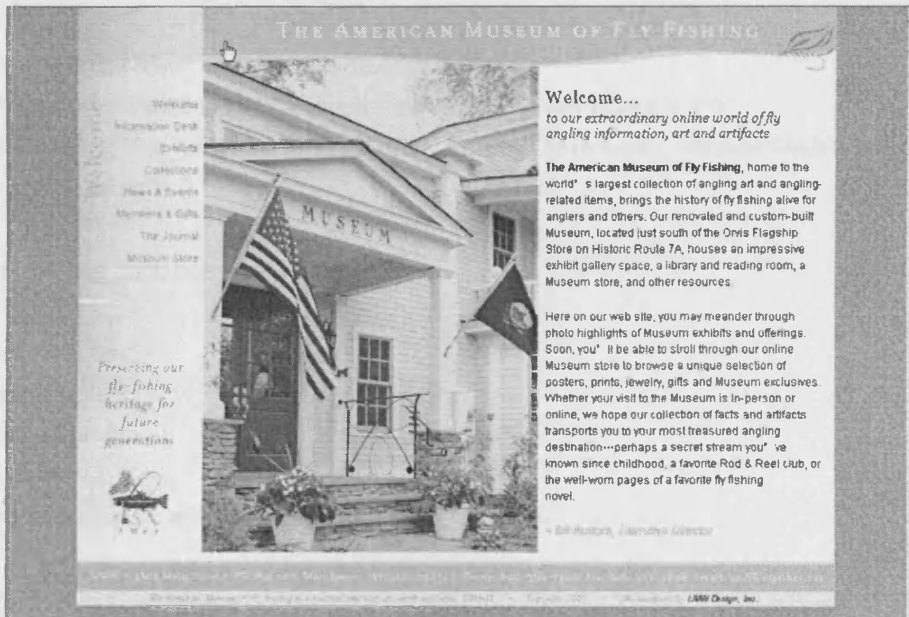


Figure A.1 Training Example 1 (a Positive Example)



Figure A.2 Training Example 2 (a Negative Example)

A.2 TRAINING EXAMPLE 1: AN HTML WEB PAGE

(FRAGMENT)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<link rel="Shortcut Icon" href="images/favicon.ico">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Welcome to The American Museum of Fly Fishing | Manchester, Vermont</title>
<style type="text/css">
<!--
@import url("mg_members_only.htm");
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
    background-image: url(images/img_102.gif);
}
.style19 {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 9px;
    line-height: 12px;
    font-weight: normal;
    font-style: italic;
    text-transform: none;
    font-variant: normal;
    text-decoration: none;
    color: #A1A582;
}
...
-->
</style>
<script language="JavaScript">
<!--
...
//-->
</script>
<script language="JavaScript1.2" src="images/popup_menu/mm_menu.js"></script>
</head>
<body ...>
...
```


A.4 SMALL DICTIONARY

Table A.1 Small Dictionary

Word #	Word	Word #	Word	Word #	Word
1	advance	62	floats	123	scale
2	alive	63	fly	124	sea
3	anchor	64	flyfishing	125	searchlight
4	anchorage	65	fresh	126	service
5	anchored	66	freshwater	127	services
6	angler	67	froth	128	set
7	anglers	68	gear	129	ship
8	attach	69	gears	130	shipping
9	attached	70	gill	131	shore
10	attaches	71	gut	132	sink
11	attaching	72	guts	133	sinker
12	backwater	73	harpoon	134	sinkers
13	bait	74	head	135	sinks
14	baits	75	hollow	136	slice
15	bamboo	76	hook	137	smoked
16	bass	77	hooks	138	smolt
17	beach	78	ice	139	spear
18	boat	79	immerse	140	spearfishing
19	boating	80	immerses	141	sport
20	bow	81	jetty	142	sports
21	bowfish	82	lake	143	steelhead
22	bowfishing	83	land	144	stern
23	box	84	landing	145	stocks
24	brailer	85	leatherworker	146	storm
25	buoy	86	line	147	stream
26	cabin	87	lure	148	streams
27	camp	88	lures	149	surround
28	carfish	89	marine	150	swim
29	carp	90	mason	151	swims
30	cast	91	mast	152	tackle
31	catch	92	metal	153	tackles
32	catfishing	93	net	154	tactics
33	caught	94	oar	155	tail
34	chop	95	ocean	156	tanner
35	choppy	96	outdoor	157	team

Table A.1 Small Dictionary (Cont.)

Word #	Word	Word #	Word	Word #	Word
36	chops	97	outdoors	158	teams
37	club	98	paddle	159	throw
38	clubs	99	pier	160	throwing
39	cobbler	100	pole	161	throws
40	competition	101	pond	162	tide
41	competitions	102	pull	163	tour
42	conversation	103	pulley	164	tournament
43	cords	104	pulling	165	tournaments
44	dale	105	pulls	166	tours
45	dip	106	punt	167	tow
46	dips	107	punts	168	trap
47	dogfish	108	pushing	169	trip
48	dried	109	recreation	170	trout
49	drown	110	release	171	tuna
50	dry	111	river	172	vans
51	entertainment	112	rod	173	vessel
52	fermented	113	rods	174	water
53	fin	114	roll	175	wave
54	fish	115	rope	176	weave
55	fisher	116	ropes	177	weaves
56	fisherman	117	row	178	weaving
57	fishermen	118	safaris	179	weight
58	fishers	119	sail	180	weights
59	fishery	120	salmon	181	woodworker
60	fishing	121	salt	182	worker
61	float	122	saltwater	183	woven

A.5 DATA FILE GENERATED BY SVM FROM THE TRAINING SET (FRAGMENT)

SVM-light Version V6.01	(1)
0 # kernel type	(2)
3 # kernel parameter -d	(3)
1 # kernel parameter -g	(4)
1 # kernel parameter -s	(5)
1 # kernel parameter -r	(6)
empty# kernel parameter -u	(7)
180 # highest feature index	(8)
62 # number of training documents	(9)
60 # number of support vectors plus 1	(10)
1.0279189 # threshold b, each following line is a SV (starting with alpha*y)	(11)
-46.429974748172633 17:0.010810811 54:0.021621622 60:0.21891892 63:0.013513514 74:0.0054054055 93:0.010810811 127:0.0027027028 141:0.0081081083 179:0.0054054055 #	(12)
...	(13)
-46.429974748172633 13:0.0090702949 30:0.0045351475 60:0.013605442 74:0.0022675737 76:0.0022675737 86:0.0022675737 102:0.0022675737 123:0.0068027209 160:0.0022675737 179:0.0022675737 #	(14)

(1) SVM^{light} Version Information.

(2) Type of kernel function:

0: Linear ($a \cdot b$)

1: Polynomial ($(\gamma \cdot a \cdot b + c)^d$)

2: Radial basis function $\exp(-\gamma \|a-b\|^2)$

3: Sigmoid $\tanh(\gamma \cdot a \cdot b + c)$

4: User defined kernel from kernel.h.

A linear kernel is used in this example.

(3) Parameter d in polynomial kernel, float type.

(4) Parameter gamma in rbf kernel, float type.

(5) Parameter s in sigmoid/poly kernel, float type.

(6) Parameter c in sigmoid/poly kernel, float type.

(7) Parameter of a user defined kernel, not used in this work, and therefore having no value.

(8) Number of features in the training examples. In this case, the total number of features used is 180.

(9) Number of training documents. In this particular example, this number is 62.

(10) In this example, there are 59 support vectors generated from the training material.

(11) The value of threshold b is generated from the training examples (see p. 8 of [Burges, 1998]). In this particular case, the value of b is 1.0279189.

(12) The first support vector generated. The first float number in it, -46.429974748172633, corresponds to $\alpha \cdot y_i$. The other float numbers in the support vector represent the features with non-zero values (see p. 9 of [Burges, 1998]).

(13) Support vectors (from no. 2 to no. 58).

(14) The last support vector generated (no. 59).

A.6 TESTING RESULTS FOR CASE STUDY 1

Table A.2 Original Output Data for Case Study 1

Web Page No.	Parameter C				
	TEST 1	TEST 2	TEST 3	TEST 4	TEST 5
63	-0.66350608	-1.1105612	-0.22711511	-0.41152923	-0.5549386
64	-0.6146509	<i>0.4459478</i>	-0.39356143	-0.3087938	-0.5100132
65	-0.75570562	<i>0.13587011</i>	-0.16377212	<i>0.43424979</i>	<i>0.1229487</i>
66	-0.88476223	<i>0.84267464</i>	-0.25979615	-0.17344767	-0.30268244
67	<i>-0.34361965</i>	0.3395136	0.37963826	0.26133312	0.08231844
68	<i>-0.56028165</i>	0.25341148	<i>-0.17184326</i>	<i>-0.15879136</i>	<i>-0.30393591</i>
69	-0.68203153	<i>0.39206489</i>	-0.17627651	<i>0.58182842</i>	-0.10495329
70	<i>-0.64461215</i>	0.044579624	<i>-0.2555542</i>	0.49053718	0.43450604
71	-0.64136793	-0.75563246	<i>0.024245796</i>	-0.24476195	-0.3837859
72	-0.54516251	-0.37603521	-0.63321543	-0.47638657	-0.63409591
73	-0.94807185	-0.61509729	-0.30822737	-0.1670702	-0.43065785
74	-0.86712238	-0.82567021	-0.8322616	-0.62191389	-0.82738891
75	-0.49105626	-0.87245945	-1.020695	-0.76788384	-0.91989471
76	<i>-0.32167472</i>	<i>-0.84650577</i>	<i>-0.44705311</i>	<i>-0.41928664</i>	<i>-0.53069734</i>
77	-0.71434127	-1.2372981	-0.84575343	-0.75068516	-0.75233001
78	<i>-0.58281349</i>	0.86997262	0.2573205	0.68869009	0.13287862
79	<i>-0.26227152</i>	<i>-0.8999278</i>	<i>-0.48396026</i>	<i>-0.98610134</i>	<i>-1.0893885</i>
80	-0.99724958	-0.7140855	-0.61744471	-0.083098426	-0.040043575
81	-0.69301305	-0.97425816	-0.38949823	-0.37589321	-0.51615454
82	-0.96153848	-0.66426467	-0.68750708	-0.22055245	-0.3796952
83	-0.31106542	-0.52663721	-0.084070847	-0.27619621	-0.37704757
84	-0.82314397	-1.0738563	-0.86472826	-0.93613782	-1.0947777
85	0.13344999	0.09906655	-0.84001762	-0.4383296	-0.71111582
86	-0.49464464	-0.56461097	-0.83911905	-0.79470071	-0.89706521
87	-0.57294458	0.57409149	0.23848707	0.95828904	0.17370597
88	-0.5877441	-0.95713128	-0.22081858	-0.19962712	-0.37811065
89	-0.51090754	-0.20023979	-0.77293411	-0.43017622	-0.55511599
90	1.1215027	1.0341855	-0.85851624	-0.44146514	-0.64543929
91	-0.33607222	0.36449767	0.4148	0.28420392	0.10584316
92	-0.75609593	-0.71717848	-0.35962196	-0.35179253	-0.45745312
93	-0.85503626	2.2094595	-0.57210336	-0.38393924	-0.63283484
94	-0.24206331	-0.02227671	1.1268569	0.27624868	0.16026111
95	-0.56628238	0.26727248	-0.17108221	-0.15644031	-0.2934512
96	1.6647492	0.6369537	1.6104216	0.43590085	0.27484654
97	1.0599881	0.80805321	-0.35893782	-0.26931719	-0.37922634
98	-0.91812616	0.44091055	-0.77983612	-0.12685412	-0.74833035
99	-0.30795662	0.76551499	-0.26128924	-0.13658279	0.017834511
100	-0.67033637	-0.67957041	-0.037781904	-0.27729066	-0.41272463

Note: Incorrect results are shown in italic.

Table A.3 Processed Results for Case Study 1

Web Page No.	Expert Opinion	TEST 1	TEST 2	TEST 3	TEST 4	TEST 5
63	N	N	N	N	N	N
64	N	N	P	N	N	N
65	N	N	P	N	P	P
66	N	N	P	N	N	N
67	P	N	P	P	P	P
68	P	N	P	N	N	N
69	N	N	P	N	P	N
70	P	N	P	N	P	P
71	N	N	N	P	N	N
72	N	N	N	N	N	N
73	N	N	N	N	N	N
74	N	N	N	N	N	N
75	N	N	N	N	N	N
76	P	N	N	N	N	N
77	N	N	N	N	N	N
78	P	N	P	P	P	P
79	P	N	N	N	N	N
80	N	N	N	N	N	N
81	N	N	N	N	N	N
82	N	N	N	N	N	N
83	P	N	N	N	N	N
84	N	N	N	N	N	N
85	N	P	P	N	N	N
86	N	N	N	N	N	N
87	P	N	P	P	P	P
88	N	N	N	N	N	N
89	N	N	N	N	N	N
90	N	P	P	N	N	N
91	P	N	P	P	P	P
92	N	N	N	N	N	N
93	P	N	P	N	N	N
94	N	N	N	P	P	P
95	P	N	P	N	N	N
96	P	P	P	P	P	P
97	P	P	P	N	N	N
98	N	N	P	N	N	N
99	P	N	P	N	N	P
100	N	N	N	N	N	N
Precision		2/4=50%	11/18=61.11%	5/7=71.43%	6/9=66.67%	7/9=77.78%
Recall		2/14=14.29%	11/14=78.57%	5/14=35.71%	6/14=42.86%	7/14=50%

P - Positive example (relevant page); N -Negative example (irrelevant page)

A.7 TESTING RESULTS FOR CASE STUDY 2

Table A.4 Original Output Data for Case Study 2

Web Page No	Parameter C	
	TEST 6	TEST 7
66	1.5100221	1.6717624
67	<i>0.89297345</i>	<i>1.1144098</i>
68	0.96766661	0.95193049
69	<i>0.15582952</i>	<i>0.017538369</i>
70	<i>-0.012009365</i>	0.073008036
71	-0.33279524	-0.059684461
72	1.6243047	1.9726233
73	-1.0936211	-1.5834467
74	0.44163406	0.56587452
75	<i>0.58372624</i>	<i>0.58503654</i>
76	0.8684108	1.0327643
77	-0.25363558	-0.34682231
78	1.1061295	1.1100589
79	-0.2883241	-0.40268144
80	0.58302483	0.7986205
81	<i>1.1890082</i>	<i>1.0013537</i>
82	0.28390861	0.11923657
83	-9.8969456	-8.9035788
84	1.384341	1.2928913
85	-0.33279833	-0.51938434

Note: Incorrect results are shown in italic.

Table A.5 Processed Results for Case Study 2

Web Page No	Expert Opinion	TEST 6	TEST 7
66	P	P	P
67	N	<i>P</i>	<i>P</i>
68	P	P	P
69	N	<i>P</i>	<i>P</i>
70	P	<i>N</i>	P
71	N	N	N
72	P	P	P
73	N	N	N
74	P	P	P
75	N	<i>P</i>	<i>P</i>
76	P	P	P
77	N	N	N
78	P	P	P
79	N	N	N
80	P	P	P
81	N	<i>P</i>	<i>P</i>
82	P	P	P
83	N	N	N
84	P	P	P
85	N	N	N
	Precision	9/13=69.23%	10/14=71.43%
	Recall	9/10=90%	10/10=100%

P - Positive example (relevant page); N -Negative example (irrelevant page)

Note: Incorrect results are shown in italic.

APPENDIX B. SAMPLE SOURCE CODE

FOR CHAPTER 4

The sample source code below illustrates a spider multithreading control class written in Java programming language.

Comments are included either

1. Between “/*” and “*/”, or
2. After “/”

```
import java.util.Vector;
import java.util.Hashtable;
import java.util.Enumeration;
import java.net.MalformedURLException;
import java.net.URL;
import java.io.*;
/**
 * Given urls from the SpiderMain class, the SpiderControl class will
 * insert and use the urls through the URLList queue and send a spider out on each
 * url to fetch and validate the webpage.  If a Webpage is valid, it will write the
 * page to a file
 */
public class SpiderControl implements Runnable
{
    /**
     * number of spiders running for spider control
     */
    public static int numOfSpiders;
    private ThreadGroup threadGroup = new ThreadGroup("spiderControl");
    private Thread currentThread = null;
    /**
     * the first urls the spiders will be searching on
     */
    private URLList urlList = new URLList();
    /**
     * keywords the spider is looking for in each webpage
     */
    private static Vector keywords;
    /**
```

```

    * the number of urls the spiders are to return
    */
private int numberUrls;
/**
    * time spider has to obtain page till it is shut off
    */
private long timeOut;
/**
    *the file path where the retrieved pages are saved to
    */
private String outputPath;
/**
    * array of spiders, each has a different thread
    */
private Spider[] spiderArray = new Spider[numOfSpiders];
Thread[] threadArray = new Thread[numOfSpiders];
private boolean[] activeArray = new boolean[numOfSpiders];
private long[] spiderRunTime = new long[numOfSpiders];
/**
    * determine the and/or choice of user
    */
private boolean or;
/**
    * determines if searchbutton is done
    */
private boolean doneSearch = false;
/**
    * number of pages retrieved with keywords
    */
private volatile int pagesRetrieved = 0;
/**
    * the index of the first page retrieved
    */
private int startPageIndex = 0;
/**
    * total number of pages retrieved with or without keywords
    */
private volatile int totalPagesRetrieved = 0;
/**
    * the SpiderMain class
    */
public SpiderMain spiderMain = null;
private long startTime = 0;
//for judging whether the target page is within the result page from searchbutton engines

```

```

private static Vector engineHosts = new Vector();
//for outputting the information to console
private LogDialog logdialog = null;
//for confirm whether the console is absent
private boolean showlog = false;
//for control the modal dialog shown when the spiders are crawling
public CrawlingDialog crawlingDialog = null;
//for make the main interface editable
public RequestKeywordsPane rKeywords = null;
/**
 * constructor - places the urls in the URLList, creates spiders, and starts thread.
 * Also constructs the directory structure that will hold all the urls collected
 */
public SpiderControl(SpiderMain parent, Vector urls, Vector keywords,
    int numberUrls, int startPageIndex, int timeOut, boolean or, LogDialog smlogD,
    boolean showlog,String outputPath, CrawlingDialog crawlingDialog,RequestKeywordsPane
rKeywords)
{
    this.spiderMain = parent;
    this.numberUrls = numberUrls;
    this.startPageIndex = startPageIndex;
    this.timeOut = (long)(timeOut * 1000);
    this.or = or;
    this.keywords = keywords;
    this.logdialog = smlogD;
    this.showlog = showlog;
    this.outputPath = outputPath;
    this.crawlingDialog = crawlingDialog;
    this.rKeywords = rKeywords;
    //add initial urls to the URLList
    this.addURLs(urls);
    this.addEngineHosts (urls);
    //run from this object's run() method
    this.currentThread = new Thread(this);
    this.currentThread.start();
    startTime = System.currentTimeMillis();
}
private void addEngineHosts (Vector initUrls)
{
    Enumeration e = null;
    if (initUrls instanceof Vector)
    {
        e = ((Vector)initUrls).elements();
    }
}

```

```

while (e.hasMoreElements())
{
    Object element = e.nextElement();
    if (element instanceof String)
    {
        try
        {
            URL tempURL = new URL((String)element);
            engineHosts.add(tempURL.getHost());
            System.out.println("addEngineHosts :"+tempURL.getHost());
            if(showlog)
                logdialog.println("addEngineHosts :"+tempURL.getHost());
        }
        catch (MalformedURLException ex) {} // end catch Malformed
    }
} //end for while (e.hasMoreElements())
}
public synchronized void addURLs(Object urls)
{
    if (getDoneSearch())
    {
        return;
    }
    // add urls to the URLList
    Enumeration e = null;
    //get it from configuration file
    if (urls instanceof Vector)
    {
        e = ((Vector)urls).elements();
    }
    //get it from the spider after analysing the pages for more links
    else if (urls instanceof Hashtable)
    {
        e = ((Hashtable)urls).elements();
    }
    else
    {
        return;
    }
    while (e.hasMoreElements())
    {
        Object element = e.nextElement();
        if (element instanceof String)
        {

```

```

        try
        {
            this.urlList.add(new ParentedURL((String)element));
        }
        catch (MalformedURLException ex) {} // end catch Malformed
    }
    else if (element instanceof ParentedURL)
    {
        this.urlList.add((ParentedURL)element);
    }
} //end for while (e.hasMoreElements())
}
/**
RUN()
loops until searchbutton is done. Checks urlList for new urls and if so, sends
a spider out with one.
*/
public void run()
{
    for (int i = 0; i < this.numOfSpiders; i++)
    {
        this.spiderArray[i] = new Spider(this,
                                        this.keywords,
                                        this.timeOut,
                                        this.or,
                                        this.engineHosts,
                                        this.logdialog,
                                        this.showlog);

        this.spiderArray[i].setSpiderNumber(i);

        this.spiderRunTime[i] = System.currentTimeMillis();
        deActiveArray(i);
        //this.threadArray[i] = new Thread(this.threadGroup, this.spiderArray[i]);
        this.threadArray[i] = (Thread)this.spiderArray[i];
        this.threadArray[i].start();
    } // end for
    ParentedURL url = this.getNextUrl();
    synchronized(this)
    {
        System.out.println("in SpiderControl run() setURL(url) "+0 +" with "+url.getURL());
        if(showlog)
        logdialog.println("in SpiderControl run() setURL(url) "+0 +" with "+url.getURL());
        this.spiderArray[0].setURL(url);
        //move the below statement to the Spider thread

```

```

        //this.activeArray[inactiveThread] = true;
    }
    while (true)
    {
        //just wait for some time to let spiders' situation changed
        this.sleep(1000);
        //get one spider is free
        int inactiveThread = this.getInactiveThread();
        // no inactive threads were found
        if (inactiveThread == -1)
        {
            continue;
        }
        url = this.getNextUrl();
        if (!this.checkDone()){
            if (url == null){
                continue;
            }
        }
        else{
            System.out.println("checkDone! spidercontrol while break");
            if(showlog)
                logdialog.println("checkDone! spidercontrol while break");
            break;
        }
        synchronized(this)
        {
            System.out.println("in SpiderControl run() setURL(url) "+inactiveThread +" with
"+url.getURL());
            if(showlog)
                logdialog.println("in SpiderControl run() setURL(url) "+inactiveThread +" with
"+url.getURL());
            this.spiderArray[inactiveThread].setURL(url);
        }
    }
} //end while (true)
//doneSearch already
this.sleep(500);
//reset activeArray all to false
for (int i = 0; i < this.numOfSpiders; i++)
{
    System.out.println("threadArray "+i+" is "+threadArray[i].isAlive());
    if(showlog)
        logdialog.println("threadArray "+i+" is "+threadArray[i].isAlive());
    deActiveArray(i);
}

```

```

} // end for
boolean active = true;
//wait until no one in the activeArray is active
while (active)
{
    active = false;
    for (int i = 0; i < this.numOfSpiders; i++)
    {
        synchronized(this)
        {
            if (this.activeArray[i])
            {
                active = true;
            }
        }
    } // end for
    if (!active)
    {
        break;
    }
}
//make all the thread stopped
for (int i = 0; i < this.numOfSpiders; i++)
{
    if (this.threadArray[i].isAlive())
    {
        System.out.println("thread "+i+" still alive");
        if(showlog)
            logdialog.println("thread "+i+" still alive");
    }
} // end for
if (this.spiderMain != null)
{
    this.spiderMain = null;
} // endif
System.out.println("SpiderControl does the job in
" +(System.currentTimeMillis()-startTime)/1000+" seconds");
if(showlog)
    logdialog.println("SpiderControl does the job in
" +(System.currentTimeMillis()-startTime)/1000+" seconds");
keywords.removeAllElements();
crawlingDialog.closeDialog();
crawlingDialog.summaryDialog();
rKeywords.enableEdit();

```



```

}
public synchronized Thread ActiveArray(int number)
{
    System.out.println("in SpiderControl ActiveArray() thread "+number);
    if(showlog)
        logdialog.println("in SpiderControl ActiveArray() thread "+number);
    this.activeArray[number] = true;
    return(this.threadArray[number]);
}
public synchronized Thread deActiveArray(int number)
{
    System.out.println("deActiveArray thread "+number);
    if(showlog)
        logdialog.println("deActiveArray thread "+number);
    this.activeArray[number] = false;
    return(this.threadArray[number]);
}
public synchronized void yieldThread(int number)
{
    this.threadArray[number].yield();
}
//this function can't be synchronized method
public void sleepThread(int number, int miliseconds)
{
    System.out.print("in SpiderControl sleepThread "+number+"\n");
    if(showlog)
        logdialog.println("in SpiderControl sleepThread "+number+"\n");
    try
    {
        this.threadArray[number].sleep(miliseconds);
        this.threadArray[number].yield();
    }
    catch (InterruptedException e) {}
}
private void sleep(int miliseconds)
{
    try
    {
        this.currentThread.sleep(miliseconds);
        this.currentThread.yield();
    }
    catch (InterruptedException e) {}
}
private int getInactiveThread()

```

```

{
    for (int i = 0; i < this.numOfSpiders; i++)
    {
        //make it maybe a better performance than enclosing
        //all the method call into a synchronized method
        synchronized(this)
        {
            if (!this.activeArray[i])
            {
                return(i);
            }
        }
    } // end for
    return(-1);
}
private ParentedURL getNextUrl()
{
    if (getDoneSearch())
    {
        return(null);
    }
    else
    {
        return(this.urlList.remove());
    }
}
private synchronized boolean checkDone()
{
    boolean active = false;
    for (int i = 0; i < this.numOfSpiders; i++)
    {
        System.out.println("in checkDone() thread "+i+" is "+threadArray[i].isAlive());
        if(showlog)
        logdialog.println("in checkDone() thread "+i+" is "+threadArray[i].isAlive());
        if (this.activeArray[i])
        {
            active = true;
            System.out.println("in checkDone,activeArray "+i+" is still active");
            if(showlog)
            logdialog.println("in checkDone,activeArray "+i+" is still active");
            //break;
        }
    }
}
System.out.println("////////////////////////////////////");

```

```

if(showlog)
logdialog.println("////////////////////////////////////////");

if (this.urlList.size() == 0 && !active)
{
    System.out.println("in checkDone (this.urlList.size() == 0 && !active)");
    if(showlog)
    logdialog.println("in checkDone (this.urlList.size() == 0 && !active)");

    setDoneSearch();
    return(true);
}
//we can't guarantee urlList is empty
//this can be improved
else if (getDoneSearch() && !active ){
    System.out.println("in checkDone getDoneSearch()"+
        "urlList.size() "+urlList.size());
    if(showlog)
    logdialog.println("in checkDone getDoneSearch()"+
        "urlList.size() "+urlList.size());
    return (true);
}
return(false);
} // end boolean checkDone()
/**
sets doneSearch to true, stopping the searchbutton
boolean value is atomized so it doesn't need sychronized
*/
public void setDoneSearch()
{
    this.doneSearch = true;
}
/**
goodHTMLPage(String, int)
This method accepts the homepages, adds to the webpage count, returns
the spider and currently write the string to a file
*/
//synchronized
public void goodHTMLPage(int spidernumber,String webpage, String location,
    String keywords, String title,
    String page, ParentedURL parent)
{
    if (getDoneSearch() || webpage.length() == 0)
    {

```

```

        System.out.println("getDoneSearch() return in goodHTMLPage, location
is :"+parent.getParentURL().getURL());
        if(showlog)
            logdialog.println("getDoneSearch() return in goodHTMLPage, location
is :"+parent.getParentURL().getURL());
        return;
    }
    //like this: http://www.dmoz.org/
    String parLoc = null;
    if (parent.getParentURL() != null)
    {
        URL url = parent.getParentURL().getURL();
        parLoc = url.toString();
        System.out.println("thread "+spidernumber+" in goodHTMLPage, location
is :"+location+" host is: "+url.getHost());
        if(showlog)
            logdialog.println("thread "+spidernumber+" in goodHTMLPage, location
is :"+location+" host is: "+url.getHost());
    } // end if(parent.getParentURL())
    else {
        System.out.println("thread "+spidernumber+" no ParentedURL, length==null!,return");
        if(showlog)
            logdialog.println("thread "+spidernumber+" no ParentedURL, length==null!,return");
        return;
    }
    // write the webpages to files
    //this synchronized method causes problems
    // if you want the output to write to a different directory,
    // set directory to something like "data\\html\\"
    String directory = outputPath + "\\";
    //this has to be synchronized to make all the pages recorded down
    synchronized (this){
        try {
            File webpageFile = new File(directory +
(this.pagesRetrieved+this.startPageIndex-1) + ".html");
            OutputStream out = new FileOutputStream( webpageFile );
            out.flush();
            String outstr = new String(location + "\n" +
"spidernumber :"+spidernumber+"\n"+webpage);
            out.write(outstr.getBytes());
            out.close();
        } catch (IOException e) {
            System.err.print("Error in writing webpages to file "+location);
            e.printStackTrace();
        }
    }
}

```

```

    } // end catch io
    catch (Exception e){
        System.out.print(e);
        //logdialog.println(e);
    }
    if (location != null){
        System.out.println(" spider "+spidernumber+" "+location+" "
            + (this.pagesRetrieved+this.startPageIndex-1)
            + ".html written to disk");
        if(showlog)
            logdialog.println(" spider "+spidernumber+" "+location+" "
                + (this.pagesRetrieved+this.startPageIndex-1)
                + ".html written to disk");
    }

    //volatile keyword in win2k doesn't take effect here
    //the recommendation from java core2 Advanced stated this clearly
    //at 53rd page volume II-advanced features gary cornell
    ++this.pagesRetrieved;
    ++this.totalPagesRetrieved;
} //synchronized (this)
if (this.pagesRetrieved >= this.numberUrls ){
    System.out.print("setDoneSearch();spiderNumber : "
        +spidernumber+
        "totalPagesRetrieved:"+totalPagesRetrieved
        +"pagesRetrieved "+pagesRetrieved );
    if(showlog)
        logdialog.println("setDoneSearch();spiderNumber : "
            +spidernumber+
            "totalPagesRetrieved:"+totalPagesRetrieved
            +"pagesRetrieved "+pagesRetrieved );
    setDoneSearch();
}
} // end goodHTMLPage(String, int)
//synchronized
public void badHTMLPage(String location, String keywords,
    String title, String page,
    ParentedURL parent)
{
    if (getDoneSearch())
    {
        System.out.println("getDoneSearch() return in badHTMLPage, location
is :"+parent.getParentURL().getURL());
        if(showlog)

```

```

        logdialog.println("getDoneSearch() return in badHTMLPage, location
is :"+parent.getParentURL().getURL());
        return;
    }
    if (location != null && location.length() > 10 )
    {
        String parLoc = null;
        if (parent.getParentURL() != null)
        {
            URL url = parent.getParentURL().getURL();
            parLoc = url.toString();
            System.out.println("in badHTMLPage, location is :"+parLoc);
            if(showlog)
                logdialog.println("in badHTMLPage, location is :"+parLoc);
        } // end if(parent.getParentURL())
        ++this.totalPagesRetrieved;
    } // end if(this.spiderMain != null)
} // end badHTMLPage
public int getPagesRetrieved()
{
    return pagesRetrieved;
} // end INT getPagesRetrieved()
public int getTotalPagesRetrieved()
{ return totalPagesRetrieved;
} // end int getTotalPagesRetrieved

// boolean value is atomized so it doesn't need sychronized
public boolean getDoneSearch()
{
    return getDoneSearch(-1);
} // end getDoneSearch()
public boolean getDoneSearch(int number)
{ if (number >0)
    System.out.println("spider "+number+" retrieve getDoneSearch");
    if(showlog)
        logdialog.println("spider "+number+" retrieve getDoneSearch");
    return(this.doneSearch);
} // end getDoneSearch()
public SpiderMain getspiderMain()
{
    return(this.spiderMain);
} // end getspiderMain()
} // end SpiderControl

```

APPENDIX C. SEMCOR, WORDNET AND ROGET'S

C.1 TAGS USED IN SEMCOR 1.6

Table C.1 Tags Used in Semcor 1.6

Tag Name	Descriptions
contextfile	Name of the semantic concordance file
context	Name of the original corpora file
p	Paragraph number in <i>context</i> ¹
s	Sentence number in <i>context</i>
wf	Syntactic and semantic information stored as attribute/value pairs
cmd	Status of the <i>wf</i> element
pos	Part of speech (noun, verb, adjective and adverb)
lemma	Root form of a word
wnsn	Word sense number in WordNet
lexsn	Lexical sense (used for indexing)
rdf	The word is redefined to something else. This is mainly used to define discontinuous collocations ² , correct typographical errors in the text, or enter a string instead of a word to search WordNet.
dc	Indicates that a word is part of a discontinuous collocation.
sep	Separator string
tagnote	Note of problems encountered during the development of Semcor
ot	Note of reasons why a tag cannot be assigned to a word
note	Additional information used in the development of Semcor

1 "Context" is used in Semcor as a tag name.

2 Discontinuous collocation is a term used in linguistics to indicate an arrangement of words which often appear at the same time but are not next to each other.

C.2 SEMCOR ANNOTATED TEXT SEMPLE

```
<contextfile concordance=brown>
<context filename=br-a01 paras=yes>
<p pnun=1>
<s snun=1>
<wf cmd=ignore pos=DT>The</wf>
<wf cmd=done rdf=group pos=NNP lemma=group wnsn=1 lexs=1:03:00::
pn=group>Fulton_County_Grand_Jury</wf>

<wf cmd=done pos=VB lemma=say wnsn=1 lexs=2:32:00::>said</wf>

<wf cmd=done pos=NN lemma=friday wnsn=1 lexs=1:28:00::>Friday</wf>
<wf cmd=ignore pos=DT>an</wf>
<wf cmd=done pos=NN lemma=investigation wnsn=1
lexs=1:09:00::>investigation</wf>
<wf cmd=ignore pos=IN>of</wf>
<wf cmd=done pos=NN lemma=atlanta wnsn=1 lexs=1:15:00::>Atlanta</wf>
<wf cmd=ignore pos=POS>'s</wf>
<wf cmd=done pos=JJ lemma=recent wnsn=2 lexs=5:00:00:past:00>recent</wf>
<wf cmd=done pos=NN lemma=primary_election wnsn=1
lexs=1:04:00::>primary_election</wf>
<wf cmd=done pos=VB lemma=produce wnsn=4 lexs=2:39:01::>produced</wf>
<punc>` `</punc>
<wf cmd=ignore pos=DT>no</wf>
<wf cmd=done pos=NN lemma=evidence wnsn=1 lexs=1:09:00::>evidence</wf>
<punc>"</punc>
<wf cmd=ignore pos=IN>that</wf>
<wf cmd=ignore pos=DT>any</wf>
<wf cmd=done pos=NN lemma=irregularity wnsn=1 lexs=1:04:00::>irregularities</wf>
<wf cmd=done pos=VB lemma=take_place wnsn=1 lexs=2:30:00::>took_place</wf>
<punc>.</punc>
</s>
</p>...
</context>
</contextfile>
```

Figure C.1 Annotated Text in Semcor (a Fragment)

C.3 WORDNET AND ROGET'S ENTRY SAMPLES

342 Land.
N. land, earth, ground, dry land, terra firma.
continent, mainland, peninsula, chersonese[Fr], delta; tongue of land, neck of land; isthmus, oasis; promontory (see projection 250); highland (see height 206).
coast, shore, scar, strand, beach; playa; bank, lea; seaboard, seaside, seacoast; ironbound coast; loom of the land; derelict; innings; alluvium, alluvion[obs3]; ancon.
riverbank, river bank, levee.
soil, glebe, clay, loam, marl, chalk, gravel, mold, subsoil, clod, clot; rock, crag.
acres; real estate (see property 780).
V. land, come to land, set foot on the soil, set foot on dry land; come ashore, go ashore, debark.
Adj. earthy, continental, midland, coastal, littoral, riparian; alluvial; terrene (see world 318); landed, territorial; rigidulous.
Adv. ashore; on shore, on land.

Figure C.2 The Entry for the Word "Land" in the Gutenberg's Edition of the Roget's Thesaurus

ABANDON^
v,
1. abandon -- (forsake, leave behind; "We abandoned the old car in the empty parking lot")
2. abandon, give up -- (give up with the intent of never claiming again; "Abandon your life to God"; "She gave up her children to her ex-husband when she moved to Tahiti"; "We gave the drowning victim up for dead")
3. vacate, empty, abandon -- (leave behind empty; move out of; "You must vacate your office by tonight")
4. abandon, give up, give -- (stop maintaining or insisting on; of ideas, claims, etc.; "He abandoned the thought of asking for her hand in marriage"; "Both sides have to give in these negotiations")
5. abandon, forsake, desolate, desert, lurch -- (leave someone who needs or counts on you; leave in the lurch; "The mother deserted her children")

Figure C.3 The Entry for the Verb "Abandon" in WordNet 1.6

APPENDIX D. SAMPLE SOURCE CODE

FOR CHAPTER 5

D.1 SAMPLE CODE FOR SEMANTIC MAPPING

The sample source code below illustrates one function in the semantic mapping programme, which is used to generate the mapping between eWord entries and OntoRo entries. The programme is written in C programming language.

Comments are included either

1. Between “/*” and “*/, or
2. After “/”

```
int findsimilarity(FILE * outputfp, conElement* conceptArray[],int conceptlen, struct wnlistitem*
wnArray[], int wnlenn, struct frequencyObj * freqlist[],int freqcount)
{
    int count1,count2;
    int freeCnt=0;//for freeing memory counter
    int currentBest =0;//current position when adding to the best points
    int listlen;//the length of the concept list for a word in wn
    int conlistNo =0; // how many conceptlist in the concept list found for a given wn list

    conElement ** conList=NULL;//for one word in wn list
    conElement* conCandidates [SEMANTICLENGTH];//holding conElement candidates of best
points, first stage, choose the best one for each word in a list
    conElement ** best;//concept list for all the members in wn for a list item,second stage, find the
best one from the best ones from the first stage

    semPoint ** semPointList = malloc(sizeof(semPoint*)*SEMANTICLENGTH);//assume there are
35 wordlist for a concept array using MACRO

    iniCandidates (conCandidates,SEMANTICLENGTH);
    best = malloc(sizeof(conElement*)*wnlenn);//array for holding all best corresponding wordnet
sense points

    for (count1=0;count1<wnlenn;count1++)//for each wn list item, find a corresponding concept line,
store the result in wnResult
    {
```



```

        best[count1]->conceptgroup,
        best[count1]->conceptno);
printf(outputfp,"%s,%d,%d,%d,%d,%d,%d,%d\n",best[count1]->expression,
        best[count1]->toplevel,
        best[count1]->section,
        best[count1]->subsection,
        best[count1]->headgroup,
        best[count1]->conceptgroup,
        best[count1]->conceptno);

//free semPointList

for(freeCnt=0;freeCnt<<((conlistNo<SEMANTICLENGTH)?conlistNo:SEMANTICLENGTH);freeC
nt++)
{
    free(semPointList[freeCnt]);
}
free(conList);
conList = NULL;
conlistNo =0;//reset the counter
} //end for

//////////free memory
free(best);
free(semPointList);
return 0;
}

```

D.2 SAMPLE CODE FOR SEMCOR ANNOTATED CORPUS PARSER

The sample source code below illustrates one function in the Semcor corpus XML format parser. This parser is used to parse Semcor corpus and convert it to OntoCorp, an ontologically tagged corpus. This programme is used after the semantic mapping between eWord and OntoRo is generated. The programme is written in Java programming language.

Comments are included either

1. Between “/*” and “*/”, or
2. After “//”

```
public void startElement(String namespaceURI,
```

```

        String lName, // local name
        String qName, // qualified name
        Attributes attrs)
throws SAXException
{
semcorNo++;
    indentLevel++;
    String eName = lName; //element
    if ("".equals(eName))
    {
        eName = qName; // namespaceAware = false //name here is<wf //should always here
        //System.out.println("+++++++startElement+++++++>" + eName + " " + lineno);
        //System.out.println(eName);
        currentElement = eName;
    }
    else {System.out.println("+++++++>" + eName + " " + lineno);}
    if (eName.length() > 0 && attrs != null && eName.equalsIgnoreCase("wf")) {
        //for everything line, lemma should be found, otherwise, the tag should be "IGNORE"
        boolean foundLemma = false;
        String keyString = new String();
        String lemmaStr = new String();
        String posStr = new String();
        int senseNo = 0;
        for (int i = 0; i < attrs.getLength(); i++) {
            String aName = attrs.getLocalName(i); // Attr name like lemma=
            if ("".equals(aName))
            {
                aName = attrs.getQName(i);
                //System.out.println("----->" + aName + " " + lineno);
            }
            //process all the attributes here, should be only one lemma string per line for :
            "<wf"

            //some precaution should be taken to guarantee there's only one per line
            if(aName.equalsIgnoreCase("lemma") &&
                !attrs.getValue(i).equals("group") &&
                !attrs.getValue(i).equals("location") &&
                !attrs.getValue(i).equals("person")){
                String temp[] = attrs.getValue(i).split(lemmaSplitSymbol);
                lemmaStr = new String(temp[0]);
                for (int j=1;j<temp.length;j++){
                    lemmaStr += " " + temp[j];
                }
                lemmaStr = lemmaStr.trim();
                foundLemma = true;

```

```

} //if
else if(aName.equalsIgnoreCase("pos")){
    posStr = attrs.getValue(i);

    //need to change the wn file senses to the n,v,a etc
    if(posStr.equalsIgnoreCase("NN")||
        posStr.equalsIgnoreCase("NNP")||
        posStr.equalsIgnoreCase("NNPS")||
        posStr.equalsIgnoreCase("NNS")||
        posStr.equalsIgnoreCase("NP")||
        posStr.equalsIgnoreCase("NPS")){
        posStr = "n";
    }
    else if(posStr.equalsIgnoreCase("JJ")){
        posStr = "a";
    }
    else if(posStr.equalsIgnoreCase("VB")){
        posStr = "v";
    }
}
else if(aName.equalsIgnoreCase("wnsn")){
    String temp = attrs.getValue(i).split(senseSplit)[0];
    senseNo = new Integer(temp).intValue();
}
/*nl();
emit("  ATTR: ");
emit(aName);
emit("\t");
emit(attrs.getValue(i));
emit("\n");*/
} //for
//make sure that lemmaStr,posStr,senseNo are all there if the wf has a recognised lemma
attribute
if(foundLemma && lemmaStr.length()>0 && posStr.length()>0 && senseNo>0){
    keyString = lemmaStr+"^"+posStr+"^"+senseNo;
    //System.out.println(">>>>>>>" +keyString+"<<<<<<<<");
    //assign the tag
    for (int i=0;i<attrs.getLength();i++){
        //now got the linked lemma string, try to find it in the similarityFile expression
        if(
            similarityMap!=null
            && similarityMap.size()>0
            && conceptMap !=null
            && conceptMap.size()>0
        ){

```

```

        if(conceptMap.containsKey(lemmaStr)){
            Vector testVector = (Vector)conceptMap.get(lemmaStr);
            if(testVector.size()==1){
                SimObj simObjTemp = (SimObj)testVector.elementAt(0);
                currentTag = "S"+String.valueOf(simObjTemp.headgroup);
                //currentTag = "****";
            }
            //can't find the best from the concept map
            else if(similarityMap.containsKey(keyString)){
                SimObj simObjTemp =
(SimObj)similarityMap.get(keyString);
                //now use headgroup as the tag to be mark for all the
words.
                currentTag =
"S"+String.valueOf(simObjTemp.headgroup);
            }
            else{currentTag = "UNKNOWN";}
        }
    }
    else {
        System.out.println("similarityMap error in 'startElement' exiting...");
        System.exit(1);
    }
} //for
}
else if(foundLemma){
    System.out.println("****No all posStr, senseNo, lemmaStr are filled,exiting...");
    System.exit(1);
}
//here should be the ignore tag now if no 'lemma' attribute found
else{
    currentTag = "IGNORE";
} //end else
} //end if
//should have a return now for each sentence;
else if(eName.length()>0 && eName.equalsIgnoreCase("s")){
    //System.out.print("in startElement 's'\n");
    sentenceCnt++;
    System.out.print("\n");
    //nl();
}
//if (attrs.getLength() > 0) nl();
//emit(">");
}

```

APPENDIX E. SAMPLE SOURCE CODE

FOR CHAPTER 6

The sample source code below illustrates one loop in the training process of the ontology tagging programme, which is used to add surrounding word counts for each word in the context of a given sense number. The programme is written in Java programming language.

Comments are included either

1. Between “/*” and “*/”, or
2. After “//”

```
for (int i = 0; i < temp.length; i++) {
    // build an entry for this concept/senseTag, with its
    // surrounding word occurrence
    if (temp[i].length() != 0) {
        String temp2[] = temp[i].split(splitSlash);
        //ensure there are only two elements
        if (temp2.length != 2) {
            System.out.println("word/Sense pair is not complete>>" +
temp2[0]
                                + "<" + temp2[1] + "<line" +
separatedlineNo+ ">>" + temp2.length
                                + "<<exiting...");
            System.exit(1);
        }
        //if ( !temp[i].equalsIgnoreCase(ignoreTag))
        if ( (!temp2[1].equalsIgnoreCase(ignoreTag))
            &&!contextMap.containsKey(temp[i].toLowerCase())) {
            Vector entry = new Vector(30);
            for (int j = 0; j != i && j < temp.length; j++) {
                //the contextMap doesn't contain the word itself, has to be
different words
                if(!temp[j].equalsIgnoreCase(temp[i])){
                    String wordtemp[] =
temp[j].toLowerCase().split(splitSlash);
                                // guarantee that there're only 2 elements after
                                // splitting, 'word' and 'senseTag',otherwise quit
                                if (wordtemp.length != 2) {
```



```

        // add to existing keys
        else if(!temp2[1].equalsIgnoreCase(ignoreTag)){
            Vector entry = (Vector)
contextMap.get(temp[i].toLowerCase());
            for (int j = 0; j != i && j < temp.length; j++) {
                String wordtemp[] =
temp[j].toLowerCase().split(splitSlash);
                // guarantee that there're only 2 elements after
                // splitting, 'word' and 'senseTag', otherwise quit
                if (wordtemp.length != 2) {
                    System.out
temp[i].toLowerCase().split(splitSlash), exiting...");
                        .println("Error in
                            System.exit(1);
                    }
                    int pos=-1;

                    //if (!temp[j].equalsIgnoreCase(ignoreTag)
                    if (!wordtemp[1].equalsIgnoreCase(ignoreTag)
                        && (pos = findPosition(entry,wordtemp[0])) >=
0) {

                            ((Sense) entry.elementAt(pos)).frequency++;
                    }
                    //didn't find wordtemp[0] in the Vector
                    //else if(!temp[j].equalsIgnoreCase(ignoreTag)){
                    else if(!wordtemp[1].equalsIgnoreCase(ignoreTag)){
                        Sense senseTemp = new Sense();
                        senseTemp.word = wordtemp[0];
                        // senseTemp.senseNo = wordtemp[1];
                        senseTemp.frequency = 1;
                        entry.add(senseTemp);
                    }//end else
                }// end for
            }// else
        }//end if (temp[i].length() != 0)
        //finished adding surrounding word frequencies to the vector list for all
the words
    }

```

APPENDIX F. SAMPLE SOURCE CODE

FOR CHAPTER 7

The sample source code below illustrates one function in the query process of the concept indexing programme, which is used to do post processing (including sequential scanning using regular expressions) according to different options supplied within queries. The query process is used to output the retrieval results from queries containing keywords, entities and concepts. The programme is written in C programming language.

Comments are included either

1. Between “/*” and “*/, or
2. After “//”

```
static int
ProcessDocs (query_data * qd, int num, int verbatim,
             char OutputType, FILE * Output)
{
    int max_buf = 0;
    int DocCount = 0;
    char *doc_sepstr = NULL;
    char *para_sepstr = NULL;
    char *para_start = NULL;
    int heads_length = atoi (GetDefEnv ("heads_length", "50"));
    char QueryType = get_query_type ();
    int need_text = (OutputType == OUTPUT_TEXT || OutputType == OUTPUT_HILITE ||
                    OutputType == OUTPUT_HEADERS || OutputType == OUTPUT_SILENT ||
                    OutputType == OUTPUT_DOCNUMS ||
                    post_proc);

    if (OutputType == OUTPUT_TEXT || OutputType == OUTPUT_HILITE)
    {
        if (QueryType == QUERY_APPROX || QueryType == QUERY_RANKED)
        {
            doc_sepstr = de_escape_string (
                Xstrdup (GetDefEnv ("ranked_doc_sepstr",
                "----- %n %w\n")));
        }
    }
    else
```

```

{
doc_sepstr = de_escape_string (
    Xstrdup (GetDefEnv ("doc_sepstr",
        "----- %n\n"));
}
para_sepstr = de_escape_string (
    Xstrdup (GetDefEnv ("para_sepstr",
        "\n##### PARAGRAPH %n #####\n"));

para_start = de_escape_string (
    Xstrdup (GetDefEnv ("para_start",
        "***** Weight = %w *****\n"));
}
if (need_text)
{
max_buf = atoi (GetDefEnv ("buffer", "1048576"));
}
do
{
u_char *UDoc = NULL;
unsigned long ULen;
if (need_text)
{
/* load the compressed text */
if (LoadCompressedText (qd, max_buf)
    {
Message ("Unable to load compressed text.");
FatalError (1, "This is probably due to lack of memory.");
}

/* uncompress the loaded text */
UDoc = GetDocText (qd, &ULen);
if (UDoc == NULL)
FatalError (1, "UDoc is unexpectedly NULL");
}
//if uncompress the text successfully and we have PostProc string in post_proc
if (!UDoc || PostProc ((char *) UDoc, verbatim))
{
fprintf(Output, "*****q d : \n");
switch (OutputType)
{
case OUTPUT_COUNT:
case OUTPUT_SILENT:
break;
}
}
}

```

```

case OUTPUT_DOCNUMS: /* This prints out the docnums string */
    if (PagerRunning)
    {
#if defined(PARADOCNUM) || defined(NZDL)
        int doc_num = GetDocNum(qd);
        if (qd->paragraph)
        {
            if (qd->id->ifh.InvflLevel == 3 &&
                (QueryType == 'R' || QueryType == 'A'))
            {
                /* Print weights for each paragraph in document */

                int true_doc_num = GetDocNumFromParaNum(qd, doc_num);

                /* Get number of paragraphs in this document */

                int num_paragraphs =
                    qd->paragraph[true_doc_num]-qd->paragraph[true_doc_num-1];

                int init_para = FetchInitialParagraph(qd->td,
                                                       doc_num);
                DocEntry *de, *doc_chain = GetDocChain(qd);
                int i;

                for (i = 0; i < num_paragraphs; i++)
                {
                    if ((de = in_chain(i, init_para, doc_chain)))
                        PrintDocNum(Output, QueryType,
                                     true_doc_num, init_para+i,
                                     de->Weight);
                }
            }
        }
        else
            PrintDocNum(Output, QueryType,
                        GetDocNumFromParaNum(qd, GetDocNum(qd)),
                        GetDocNum(qd),
                        GetDocWeight(qd));
    }
#else
    {
        PrintDocNum(Output, QueryType,
                    doc_num, doc_num, GetDocWeight(qd));
    }
#endif
}
#endif

```

```

fprintf(Output, " *****\n");
//output only the document number
fprintf (Output, "%7d   %6.4f   %7lu\n", GetDocNum (qd),
        GetDocWeight (qd), GetDocCompLength (qd));
#endif
}
break;
case OUTPUT_HEADERS: /* This prints out the headers of the documents */
    if (PagerRunning)
        fprintf (Output, "%d ", GetDocNum (qd));
        HeaderOut (Output, UDoc, ULen, heads_length);
        if (PagerRunning)
            fprintf ("\\n", Output);
            break;
#if TREC_MODE
case OUTPUT_EXTRAS:/* This prints out the docnums string */
    if (PagerRunning && trec_ids)
    {
        long DN, PN = GetDocNum (qd) - 1;
        if (trec_paras)
            DN = trec_paras[PN];
        else
            DN = PN;
        fprintf (Output, "%-14.14s   %8ld   %10.5fn",
                &trec_ids[DN * 14], PN + 1, GetDocWeight (qd));
    }
    break;
#endif
case OUTPUT_TEXT:
case OUTPUT_HILITE:
    {
        int j, para = -1, curr_para = 0;
        int init_para = -1;
        DocEntry *de, *doc_chain = NULL;
        register char ch = ' ';
        register char lch = '\\n';
#if defined(PARADOCNUM) || defined(NZDL)
        if (qd->id->ifh.InvflLevel == 3)
        {
            init_para = FetchInitialParagraph(qd->td, GetDocNum(qd));

            StringOut(Output, para_sepstr,
                    1, init_para+curr_para,
                    0, 0);

```

```

    }
else
    StringOut(Output, doc_sepstr,
              1, GetDocNum(qd),
              QueryType == 'A' || QueryType == 'R',
              GetDocWeight(qd));
#else
int p_on = 0;

if (PagerRunning)
{
    StringOut (Output, doc_sepstr,
              1, GetDocNum (qd),
              QueryType == 'A' || QueryType == 'R',
              GetDocWeight (qd));
}
if (qd->id->ifh.InvfLevel == 3)
{
    init_para = FetchInitialParagraph (qd->td, GetDocNum (qd));
    doc_chain = GetDocChain (qd);
    para = GetDocNum (qd) - init_para;
    StringOut (Output, para_sepstr,
              1, curr_para + 1,
              0, 0);
    if ((de = in_chain (0, init_para, doc_chain)))
        StringOut (Output, para_start,
                  0, 0,
                  1, de->Weight);
    if (doc_chain->DocNum - init_para == 0)
        p_on = 1;
}
#endif

for (j = 0; j < ULen; j++)
{
    ch = UDoc[j];
    switch (ch)
    {
        case '\02':
            break;
        case '\01':
            ch = '\n';
        case '\03':
            break;
    }
}
#endif
/* print paragraph numbers only if this is

```

```

        a level 3 index */
    if (qd->id->ifh.InvfLevel == 3)
    {
        curr_para++;
        StringOut(Output, para_sepstr,
            1, init_para+curr_para,
            0, 0);
    }
#else
    p_on = 0;
    curr_para++;
    StringOut (Output, para_sepstr,
        1, curr_para + 1,
        0, 0);
    lch = *(strchr (para_sepstr, '\0') - 1);
    if ((de = in_chain (curr_para, init_para, doc_chain))
        StringOut (Output, para_start,
            0, 0,
            1, de->Weight);
    if (doc_chain &&
        doc_chain->DocNum - init_para == curr_para)
        p_on = 1;
#endif
    break;
    default:
    {
        if (PagerRunning)
        {
            fputc (ch, Output);
            #if !defined(PARADOCNUM) && !defined(NZDL)
                if (p_on && isprint (ch))
                {
                    fputc ('\b', Output);
                    fputc ('_', Output);
                }
            #endif
        }
        #endif

        lch = ch;
    }
}
}
}
if (PagerRunning && lch != '\n')
    fputc ('\n', Output);

```



```

#if !defined(PARADOCNUM) && !defined(NZDL)
    p_on = 0;
#endif
    }
    }
    if (PagerRunning)
        fflush (Output);
    DocCount++;    }
}

while (NextDoc (qd) && PagerRunning && (!Ctrl_C));
    //removing previously allocated re* pointer resource
    if(re!=NULL)
    {
        regfree(re);
        free(re);
        re = NULL;
    }
if (need_text)
{
    FreeTextBuffer (qd);
}

if (OutputType == OUTPUT_TEXT || OutputType == OUTPUT_HILITE)
{
    Xfree (doc_sepstr);
    Xfree (para_sepstr);
    Xfree (para_start);
}
return (DocCount);
}

```

APPENDIX G. DATA OF TESTING RESULTS USED IN

CHAPTER 7

G.1 INDEXING TESTS

Table G.1 Total Processing Time for Entity Indexing

Total Processing Time, secs	Words, million
229	6
499	12
664	18
870	24
2140	30
2448	36
1688	42
2983	48
2065	54
2252	60

Table G.2 Total Processing Time for Concept Indexing

Total Processing Time, secs	Words, million
71	6
142	12
209	18
279	24
354	30
431	36
487	42
559	48
625	54
701	60

Table G.3 Total Processing Time for Merged Indexing

Total Processing Time, secs	Words, million
52	6
84	12
132	18
220	24
268	30
336	36
384	42
417	48
390	54
429	60

G.2 TESTING OF ENTITY AND CONCEPT EXTRACTION

Table G.4 Tests of Entity Extraction

CPU Processing Time, secs	Total Processing Time, secs	Disk Usage, KB	Memory Usage, KB	Words, million
72.5	76	11373.1	737.1	6
144.7	149	22739	772.4	12
218	224	34110.8	805.8	18
289.4	291	45490.6	838.7	24
362	369	56861.9	870.9	30
433.9	474	68233.5	907.1	36
502.2	507	79604.6	939.7	42
573.8	581	90984.8	972.9	48
645.6	653	102356.8	1003.8	54
716.7	724	113728.5	1034.6	60

Table G.5 Tests of Concept Extraction

CPU Processing Time, secs	Total Processing Time, secs	Disk Usage, KB	Memory Usage, KB	Words, million
11.5	13	2838.6	351.4	6
22.7	22.7	5650.9	359	12
34.3	37	8466.3	367.3	18
45.8	47	11281.6	375.8	24
56.9	70	14097	383.7	30
68.9	74	16912.3	393.3	36
79.6	83	19727.8	401.3	42
91	95	22543.2	409.6	48
103	110	25358.6	417.7	54
114.4	122	28173.9	426.2	60

G.3 RETRIEVAL SPEED TESTS

Table G.6 Retrieval Speed Tests between MySQL and the Developed System

Test Number	Retrieving Time of MySQL, ms	Retrieving Time of the Developed System, ms
1	4053.5	82.5
2	555	196
3	156.2	78
4	120.2	56
5	66	38
6	116.2	30
7	52.4	34
8	66	30
9	47.5	20
10	45	25

REFERENCES

Allan, J., Aslam J., Belkin, N., Buckley, C., Callan, J., Croft, B., Dumais, S., Fuhr, N., Harman, D., Harper, D. J., Hiemstra D., Hofmann, T., Hovy, E., Kraaij, W., Lafferty, J., Lavrenko, V., Lewis, D., Liddy, L., Manmatha, R., McCallum, A., Ponte, J., Prager, J., Radev, D., Resnik, P., Robertson, S., Rosenfeld, R., Roukos, S., Sanderson, M., Schwartz, R., Singhal, A., Smeaton, A., Turtle, H., Voorhees, E., Weischedel, R., Xu, J. and Zhai, C. (2003). Challenges in Information Retrieval and Language Modelling: Report of A Workshop Held at The Centre for Intelligent Information Retrieval, The ACM SIGIR Forum, 37(1), pp. 31-47.

American National Standards Institute (ANSI), (1984). American National Standard for Library and Information Sciences and Related Publishing Practices - Basic Criteria for Indexes (Z39.4-1984), American National Standards Institute, USA.

Antoniou, G. and van Harmelen, F. (2004). A Semantic Web Primer, The MIT Press, UK.

Araujo, M., Navarro, G. and Ziviani, N. (1997). Large Text Searching Allowing Errors, in Proc. of WSP'97, Valparaiso, Chile, pp. 2-20.

Arjona, J. L., Corchuelo, R. and Toro, M. (2003). A Knowledge Extraction Process Specification for Today's Non-Semantic Web, in Proc. of IEEE/WIC International Conference on Web Intelligence (WI 2003), Halifax, Canada, pp. 61-67.

Avello, D. G., Gutierrez, D. I. and Lovelle, J. M. C. (2002). A Concept-Based Retrieval Tool: The Cooperative Web, in Proc. of IADIS International Conference WWW/Internet (ICWI 2002), Lisbon, Portugal, pp. 712-715.

Baeza-Yates, R. (2003). Information Retrieval in the Web: Beyond Current Search Engines, *International Journal of Approximate Reasoning*, 34(2), pp. 97-104.

Baeza-Yates, R. and Navarro, G. (1999). Faster Approximate String Matching, *Algorithmica*, 23(2), pp.127-158.

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*, Addison Wesley, USA.

Beckman, T. (1997). A Methodology for Knowledge Management, *International Association of Science and Technology For Development (IASTFD) AI and Soft Computing Conference*, Banff, Canada.

Benjamins, V. R., Contreras, J. and Gomez-Perez, A. (2002). Six Challenges for the Semantic Web. in Proc. of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002), Semantic Web Workshop, Toulouse, France.

Berners-Lee, T., Hendler, J. and Lassila, O. (2001). The Semantic Web. *Scientific American*, 28, pp. 34-43.

Blackler, F. (1995). Knowledge, Knowledge Work and Organizations: an Overview

and Interpretation, *Organization Studies*, 15(6), pp. 1021-1046.

Boicu, M., Tecuci, G., Stanescu, B., Balan, G. C. and Popovici, E. (2001). Ontologies and the Knowledge Acquisition Bottleneck, in Proc. of IJCAI 2001, Workshop on Ontologies and Information Sharing, pp. 9-18. Seattle, USA.

Bontcheva, K., Tablan, V., Maynard, D., Cunningham, H. (2004). Evolving GATE to Meet New Challenges in Language Engineering, *Natural Language Engineering*, 10 (3/4), pp. 349-373.

Booch, G., Jacobson, I., and Rumbaugh, J. (1998). *The Unified Modeling Language User Guide*, Addison-Wesley, USA

Boser, B. E., Guyon, I. M. and Vapnik, V. N. (1992). A Training Algorithm for Optimal Margin Classifiers. in Proc. of 5th Annual ACM Workshop on COLT, Haussler D.(Ed.), Pittsburgh, USA, pp. 144-152.

Bou, B. (2005). <http://wnjn.sourceforge.net>, [Accessed on 16 Jan, 2006].

Brachman, R. J. and Schmolze, J. (1985). An Overview of the KL-ONE Knowledge Representation System, *Cognitive Science*, 9(2), pp. 171-216.

Brewington, B. E. and Cybenko, G. (2000). Keeping Up with the Changing Web, *IEEE Computer*, 33(5), pp. 52-58.

Brickley, D. and Guha, R.V. (2004). Resource Description Framework Schema

Specification 1.0, Technical Report, The World Wide Web Consortium (W3C), USA.
<http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>, [Accessed on 19 Feb, 2006].

Brill, E. (1992). A Simple Rule-Based Part of Speech Tagger, in Proc. of 3rd Conference on Applied Natural Language Processing (ANLP92'), Trento, Italy, pp.152-155.

Broder, A. Z. and Ciccolo, A. C. (2004). Towards the Next Generation of Enterprise Search Technology, IBM Systems Journal, 43(3), pp. 451-454.

Browne, G. M. (2003). Automatic Categorisation. Part 1: Principles of Classification, Online Currents, 18(1), pp. 17-22.

Burges, C. J. C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 2(2), pp. 121-167.

Chakrabarti, S. (2002). Mining the Web: Discovering Knowledge from Hypertext Data, Morgan Kaufmann, USA.

Chakrabarti, S., van den Berg, M., and Dom, B. (1999). Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery, Computer Networks, 31(11-16), pp. 1623-1640.

Chang, C. C. and Lin, C. J. (2002). Training Nu-Support Vector Regression: Theory and Algorithms, Neural Computation, 14, pp. 1959-1977.

Chang, C. T. K. and Schatz, B. R. (1999). Performance and Implications of Semantic Indexing in a Distributed Environment, in Proc. of the 8th International Conference on Information and Knowledge Management, Kansas City, USA, pp. 391-398.

Chen, H. and Dumais, S. (2000). Bringing Order to the Web: Automatically Categorizing Search Results, in Proc. of the SIGCHI Conference on Human Factors in Computing Systems, The Hague, Netherlands, pp. 145-152.

Chen, P. P. (1976). The Entity-Relationship Model - Toward a Unified View of Data, ACM Transactions on Database Systems (TODS), 1(1), pp. 9-36.

Cho, J. and Garcia-Molina, H. (2000). The Evolution of the Web and Implications for an Incremental Crawler, in Proc. of 26th International Conference on Very Large Databases (VLDB 2000), Cairo, Egypt.

Cho, J., Garcia-Molina, H. and Page, L. (1998). Efficient Crawling Through URL Ordering, in Proc. of 7th World Wide Web Conference (WWW7), Brisbane, Australia.

Ciravegna, F. and Wilks, Y. (2003). Adaptive Information Extraction for the Semantic Web in Amilcare, in Annotation for the Semantic Web, Handschuh, S. and Staab, S. (Eds.), IOS Press, Netherlands.

Cooper, W. S. (1994). The Formalism of Probability Theory in IR: A foundation or An Encumbrance? in Proc. of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland, pp. 242-247.

Corcho, O. and Gomez-Perez, A. (2000). A Road Map to Ontology Specification Languages, in Proc. of the 12th International Conference on Knowledge Acquisition, Modeling and Management, Juan-les-Pins, France. pp. 80-96.

Corcho, O., Fernandez-Lopez, M. and Gomez-Perez, A. (2003). Methodologies, Tools and Languages for Building Ontologies: Where is Their Meeting Point? Data Knowledge Engineering. 46(1), pp. 41-64.

Crestani, F., Lalmas, M., van Rijsbergen, C. J., and Campbell, I. (1998). Is This Document Relevant?? - Probably: a Survey of Probabilistic Models in Information Retrieval, ACM Computing Surveys (CSUR), 30(4), pp. 528-552.

Cunningham, H., Maynard, D., Bontcheva, K. and Tablan, V. (2002). GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications, in Proc. of the 40th Anniversary Meeting of the Association for Computational Linguistics, Philadelphia, USA.

Daelemans, W. and van den Bosch, A, (2005). Memory-Based Language Processing, Cambridge University Press, UK.

Daelemans, W., Zavrel, J., van der Sloot, K., and van den Bosch, A. (2004). TiMBL: Tilburg Memory Based Learner, version 5.1, Reference Guide, ILK Research Group Technical Report Series No. 04-02, 2004. ILK Pub: ILK-0402, Tilburg University, Netherlands.

Davies, J., Duke, A., Kings, N., Mladenic, D., Bontcheva, K., Grcar, M., Benjamins, R., Contreras, J., Civico, M. B., Glover, T. (2005). Next Generation Knowledge Access, *Journal of Knowledge Management*, 9(5), pp. 64-84.

Davies, J., Studer, R., Sure, Y. and Warren, P. (2005). Next Generation Knowledge Management, *BT Technology Journal* 23 (3), pp. 175-190.

Davies, N. J. (2000). Knowledge Management, *BT Technology Journal*, 18(1), pp. 62-63.

Davies, N. J. (2003). *Towards the Semantic Web*, John Wiley & Sons Ltd., UK.

Davis, R., Shrobe, H. and P. Szolovits. (1993). What is a Knowledge Representation, *AI Magazine*, 14(1), pp. 17-33.

Devitt, A. and Vogel, C. (2004). The Topology of WordNet: Some Metrics. In *Proc. of the Second Global WordNet Conference*, Brno, Czech Republic, pp. 106-111.

Dovey, K. (1997). The Learning Organisation and Organisation of Learning - Power, Transformation and the Search for Form in Learning Organisations, *Management Learning*, 28(3), pp. 331-349.

Dretske, F.I. (1981). *Knowledge and the Flow of Information*, MIT Press, USA.

Dumais, S. (1998). Using SVMs for Text Categorisation, *IEEE Intelligent Systems*, 13(4), pp. 21-23

Eckel, B. (2000). *Thinking in C++* (2nd Ed.), Prentice Hall, USA.

Eschenfelder, E., Heckman, R. and Sawyer, S. (1998). The Distribution of Computing: the Knowledge Markets of Distributed Technical Support Specialists, *Information Technology and People*, 11(2), pp. 84-103.

Fellbaum, C. (Ed.), (1998). *WordNet: An Electronic Lexical Database*. The MIT Press, USA.

Fensel, D. (Ed.), (2002). *Spinning the Semantic Web: Bring the World Wide Web to Its Full Potential*, The MIT Press, USA.

Francis, W. and Kucera, H. (1979). *Brown Corpus Manual*, Department of Linguistics, Brown University, USA.

Free Software Foundation, Inc. (FSF), (2006). <http://www.gnu.org/software/gawk>, [Accessed on 16 Jan, 2006].

Fuhr, N. (1992). Probabilistic Models in Information Retrieval. *The Computer Journal*, 35(3), pp. 243-255.

Furnas, G. W., Deerwester, S., Dumais, S. T., Landauer, T. K., Harshman, R.A., Streeter, L. A. and Lochbaum, K. E. (1988). Information Retrieval Using a Singular Value Decomposition Model of Latent Semantic Structure, in *Proc. of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Grenoble, France, pp. 465-480.

Garside, R., Leech, G. and McEnery, A. (1997). *Corpus Annotation: Linguistic Information from Computer Text Corpora*, Addison Wesley, UK.

Glover, E. J., Tsioutsoulouklis, K., Lawrence, S., Pennock, D. M. and Flake, G. W. (2002). Using Web Structure For Classifying And Describing Web Pages, in Proc. of the 11th International Conference on World Wide Web (WWW2002), Honolulu, USA, pp. 562-569.

Gottgroy, P., Kasabov, N. and Macdonell, S. (2003). An Ontology Engineering Approach for Knowledge Discovery from Data in Evolving Domains, in Proc. of Data Mining 2003, Rio de Janeiro, Brazil.

Griffith, J. and O'Riordan, C. (2003). A Formal Framework for Combining Evidence in an Information Retrieval Domain. in *Knowledge-Based Intelligent Information and Engineering Systems*, Palade, V., Howlett, R. J. and Jain, L. (Eds.), *Lecture Notes in AI*, 2773, pp. 864-871.

Grishman, R. (2003). Information Extraction, in *The Oxford Handbook of Computational Linguistics*, Mitkov, R. (Ed.), Oxford University Press, Oxford, UK, pp. 545-559.

Harman, D., Fox, E., Baeza-Yates, R. and Lee, W. (1992). Inverted Files. in *Information Retrieval: Algorithms and Data Structures*, Frakes, W. and Baeza-Yates, R. (Eds.), pp. 28-43. Prentice Hall, USA.

Henstock, P. V., Pack, D. J., Lee, Y. and Weinstein, C. J. (2001). Toward an Improved Concept-Based Information Retrieval System, in Proc. of 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New Orleans, USA.

Heydon A. and Najork, M. (1999). Mercator: A Scalable, Extensible Web Crawler, in Proc. of the 8th World Wide Web Conference (WWW8), 2(4), pp. 219-229.

Holub, M. (2003). A New Approach to Conceptual Document Indexing: Building a Hierarchical System of Concepts Based on Document Clusters, in Proc. of the 1st International Symposium on Information and Communication Technologies, Dublin, Ireland., pp. 310-315.

Hopcroft, J. E., Motwani, R. and Ullman, J. D. (1992). Introduction to Automata Theory, Languages, and Computation, (2nd Ed.) Addison-Wesley, USA.

Horrocks, I., Patel-Schneider, P. and Harmelen., F. (2002). Reviewing the Design of DAML+OIL: An Ontology Language for the Semantic Web. Technical Report, Defense Advanced Research Projects Agency.

Hou, J. L., Sun, M. T. and Chuo, H. C. (2005). An Intelligent Knowledge Management Model for Construction And Reuse of Automobile Manufacturing Intellectual Properties, International Journal of Advanced Manufacturing Technology, pp. 169-182.

Huffman, D. (1952). A Method for the Construction of Minimum-Redundancy Codes, in Proc. of the I.R.E., 40(9), pp. 1090-1101.

IDM Computer Solutions, Inc. (IDMCS), (2005). <http://www.ultraedit.com>, [Accessed on 16 Jan, 2006].

Isozaki, H. (2001). Japanese Named Entity Recognition Based on a Simple Rule Generator and Decision Tree Learning. ACL 2001, pp. 306-313.

Jackson, I. (2006). <http://www.chiark.greenend.org.uk/~ian/adns/>, [Accessed on 20 Feb, 2006].

Jacobs, P. S. and Rau, L. (1990). SCISOR: Extracting information from On-Line News. Communications of the ACM, 33(11), pp. 88-97.

Joachims, T. (1998). Making Large-Scale SVM Learning Practical, MIT Press, USA.

Joachims, T. (1998). Text Categorisation with Support Vector Machines: Learning with Many Relevant Features. LS-8 Report 23, University of Dortmund, Computer Science Department, Germany.

Joachims, T. (2001). A Statistical Learning Model of Text Classification for Support Vector Machines. SIGIR Forum, pp. 128-136.

Joachims, T. (2002). Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms, Springer, Germany.

Joy, B., Steele, G. Bracha, G and Gosling, G. (2005). The Java Language Sepcification, Addison Wesley, USA.

Jurisica, I., Mylopoulos, J. and Yu, E. (2004). Ontologies for Knowledge Management: An Information Systems Perspective, Knowledge and Information Systems, 6(4), pp. 380-401.

Kakabadse, N. K., Kakabadse, A. and Kouzmin A. (2003). Reviewing the Knowledge Management Literature: Towards a Taxonomy, Journal of Knowledge Management, 7(4), pp. 75-91.

Karypis, G. and Han, E. H. (2000). Concept Indexing: A Fast Dimensionality Reduction Algorithm with Applications to Document Retrieval and Categorization, in Proc. of 9th International Conference on Information and Knowledge Management (CIKM 2000), Washington, USA. pp. 12-19.

Klyne, G. and Carroll, J. (Eds.) (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax, <http://www.w3.org/TR/rdf-concepts>

Kobayashi, M. and K. Takeda, Information Retrieval on the Web, ACM Computing Surveys 32(2), pp. 144 - 173.

Kwok, K. L. (1995). A Network Approach To Probabilistic Information Retrieval. ACM Transactions on Information Systems. 13(3), pp. 324-353.

Lave, J. (1988). Cognition in Practice, Cambridge University Press, USA.

Lee, J. H., Kim, W. Y. and Lee, Y. H. (1993). Ranking Documents in Thesaurus-based Boolean Retrieval Systems. *Information Processing & Management*, 30(1), pp. 79-91

Li, X. L. and Liu, J. M. (2001). Chinese Web Page Classifier Based on Support Vector Machine and Unsupervised Clustering, *Jisuanji Xuebao*, Chinese Journal of Computers 24(1). pp. 62-68.

Luke, S., Spector, L., Rager, D. and Hendler, J. (1997). Ontology-based Web Agents, in Proc. of the First International Conference on Autonomous Agents (Agents'97), Johnson W. and Hayes-Roth, B. (Eds.), Marina del Rey, USA, pp. 59-68.

Maedche, A. and Staab, S. (2000). Discovering Conceptual Relations from Text, in Proc. of the 14th European Conference on Artificial Intelligence (ECAI 2000), Berlin, Germany.

Manber, U. and Wu, S. Glimpse: A tool to Search Through Entire File Systems. Tech. Report 93-34, Department of Computer Science, University of Arizona.

Manber, U., Smith, M. and Gopal, B. (1997). WebGlimpse - Combining Browsing and Searching, Usenix Technical Conference, Anaheim, USA.

Marwick, A. D. (2001). Knowledge Management Technology, *IBM System Journal*, 40(4), pp. 814-830.

Matsumoto, Y. (2003). Lexical Knowledge Acquisition, in *The Oxford Handbook of Computational Linguistics*, Mitkov, R. (Ed.), Oxford University Press, Oxford, UK,

pp 395-413.

Maynard, D., Bontcheva, K. and Cunningham, H. (2003). Towards a Semantic Extraction of Named Entities, in Proc. of Recent Advances in Natural Language Processing, Borovets, Bulgaria.

McGuinness, D. L. and van Harmelen, F. (Eds.) (2004). OWL Web Ontology Language, <http://www.w3.org/TR/owl-features/>

Mihalcea, R. and Moldovan, D. (2000). Semantic Indexing Using WordNet Senses, in Proc. of 38th Annual Meeting of the Association for Computational Linguistics (ACL 2000), Workshop on Recent Advances in NLP and IR, Hong Kong, China.

Miller, G. A. (1990). WORDNET: An On-Line Lexical Database, International Journal of Lexicography, 3(4), pp. 235-312.

Minnen, G., Carroll, J. and Pearce, D. (2001). Applied Morphological Processing of English, Natural Language Engineering, 7(3), pp. 207-223.

Minsky, M. (1975). A Framework for Representing Knowledge. in the Psychology of Computer Vision, Winston, P. (Ed.), McGraw-Hill, pp. 211-277, USA

Mitkov, R. (Ed.), (2003). The Oxford Handbook of Computational Linguistics, Oxford University Press, Oxford, UK.

Miyamoto, S., Miyake, T. and Nakayama, K. (1983). Generation of a Pseudothesaurus

for Information Retrieval Based on Co-occurrences and Fuzzy Set Operations, IEEE Transactions on Systems and Man Cybernetics, 13(1), pp. 62-70.

Moolenaar, B. (2005). <http://www.vim.org>, [Accessed on 16 Jan, 2006].

Mooney, R. J. (2003). Machine Learning, in The Oxford Handbook of Computational Linguistics, Mitkov, R. (Ed.), Oxford University Press, Oxford, UK, pp. 545-559.

Mostafa, J., Mukhopadhyay, S., Lam, W. and Palakal, M. (1997). A Multilevel Approach to Intelligent Information Filtering: Model, System and Evaluation, ACM Transactions on Information Systems, pp. 368-369.

Moura, E., Navarro, G., Ziviani, N. and Baeza-Yates, R. (1998). Fast Searching on Compressed Text Allowing Errors, in Proc. of SIGIR'98, pp. 298-306.

Mukherjee, R. and Mao, J. (2004). Enterprise Search: Tough Stuff, Queue, 2(2), pp. 36-46.

Nichols, B., Buttlar, D. and Farrell, J. P. (1996). Pthreads Programming, O'Reilly and Associates, USA.

Nonaka, I. (1994). A Dynamic Theory of Organizational Knowledge Creation, Organization Science, 5(1), pp. 14-37.

Nonaka, I., Toyama, R. and Konno, N. (2000). SECI, Ba and Leadership: A Unified Model of Dynamic Knowledge Creation, Long Range Planning, 33, pp. 5-34.

Noy, N. F., Sintek, M., Decker, S., Crubezy, M., Ferguson, R. W. and Musen, M. A. (2001). Creating Semantic Web Contents with Protege-2000. *IEEE Intelligent Systems* 16(2), pp. 60-71.

O'Dell, C. and Jackson, C. (1998). *If Only We Know What We Know: The Transfer of Internal Knowledge and Best Practice*, Free Press, USA

Pham, D. T. and Karaboga, D. (2000). *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*, Springer, UK.

Pham, D. T. and Xing, L. (1995). *Neural Networks for Identification, Prediction, and Control*, Springer-Verlag, UK.

Plato. (1953). *Phaedo*, in *Plato I*, Gowler, H.N. (Ed. and Trans.), Harvard University Press USA, pp. 117-24.

Popov, B., Kiryakov, A., Manov, D., Kirilov, A., Ognyanoff, D. and Goranov, M. (2003). Towards Semantic Web Information Extraction, in *Proc. of 7th IEEE International Symposium on Wearable Computers (ISWC'03) Workshop on Human Language Technology for the Semantic Web and Web Services*, New York, USA.

Princeton University. (2006). <http://wordnet.princeton.edu/papers.shtml>, [Accessed on 16 Jan, 2006].

Priss, U. (2000). Faceted Information Representation, *Electronic Transactions on Artificial Intelligence*, 4, pp. 21-33.

Project Gutenberg Literary Archive Foundation, (PGLAF). (2005).
<http://www.gutenberg.org>, [Accessed on 16 Jan, 2006].

Quaddus, M. and Xu, J. (2005). Adoption and Diffusion of Knowledge Management Systems: Field Studies of Factors and Variables, *Knowledge-Based Systems*, 18, pp. 107-115.

Quillian, M. R. (1968). Semantic Memory, in *Semantic Information Processing*, Minsky, M. (Ed.), pp. 216-270, MIT Press, USA.

Radecki, T. (1976). Mathematical Model of Information Retrieval System Based on the Concept of Fuzzy Thesaurus. *Information Processing & Management*, 12, pp. 313-318.

Radecki, T. (1982). Incorporation of Relevance Feedback into Boolean Retrieval Systems, *SIGIR 1982*, West Berlin, Germany, pp. 133-150.

Ribeiro-Neto, B. A. and Muntz, R. A Belief Network Model for IR, in *Proc. of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Zurich, Switzerland, pp. 253-260.

Roget, P. (2003). *Roget's Thesaurus of English Words and Phrases*. Davidson, G. (Ed.), Penguin Books Ltd, UK.

S. Brin and L. Page. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine, in *Proc. of the 7th World Wide Web Conference (WWW7)*, Brisbane,

Australia.

Salton, G. (1971). *The SMART Retrieval System - Experiments in Automatic Document Processing*, Prentice Hall, USA.

Salton, G. and Buckley, C. (1988). Term - Weighting Approaches in Automatic Retrieval, *Information Processing and Management*, 24(5), pp. 513-523.

Salton, G. and Lesk, M. E. (1968). Computer Evaluation of Indexing and Text Processing, *Journal of the ACM*, 15(1), pp. 8-36.

Salton, G., Fox, E. A. and Wu, H. (1982). Extended Boolean Information Retrieval, *Communications of the ACM*, 26(11), pp. 1022-1036.

Samuelsson, C. (2003). Statistical Methods, in *The Oxford Handbook of Computational Linguistics*, Mitkov, R. (Ed.), Oxford University Press, Oxford, UK, pp 358-375.

Scholkopf, B. and Smola, A. J. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, USA.

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization, *ACM Computing Surveys (CSUR)*, 34(1), pp. 1-47.

Shannon, C. E. (1948). A Mathematical Theory of Communication, *Bell System Technical Journal*, 27, pp. 379-423.

Stevens, W. R. (1994). *TCP/IP Illustrated: The Protocols v. 1 (APC)*, Addison Wesley, UK.

Stevenson, M. (2004). Information Extraction from Single and Multiple Sentences, in *Proc. of the 20th International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland, pp. 875-881.

Stevenson, M. and Ciravegna, F. (2003). Information Extraction as a Semantic Web Technology: Requirements and Promises. *Adaptive Text Extraction and Mining Workshop at the 14th European Conference on Machine Learning (ECML 2003)*, Cavtat-Dubrovnik, Croatia.

Stevenson, M. and Wilks, Y. (2003). Word-Sense Disambiguation, in *The Oxford Handbook of Computational Linguistics*, Mitkov, R. (Ed.), Oxford University Press, UK.

Stokeo, C., Oakes, M. and Tait, J. (2003). Word Sense Disambiguation in Information Retrieval Revisited, in *Proc. of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Toronto, Canada.

Stokoe, C., Oakes, M. P. and Tait, J. (2003). Word Sense Disambiguation in Information Retrieval Revisited, in *Proc. of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Toronto, Canada, pp. 159-166.

The Apache Software Foundation, (ASF). (2005). <http://xml.apache.org>, [Accessed on 16 Jan, 2006].

The University of Sheffield, (2006). <http://gate.ac.uk/annie/index.jsp>, [Accessed on 19 Feb, 2006].

Thede, S. M. and Harper, M. P. (1999). A Second-Order Hidden Markov Model for Part-of-Speech Tagging, in Proc. of the 37th Annual Meeting of the Association for Computational Linguistics (ACL99'), pp. 175-182.

Tudhope, D. (2004). A Case Study of a Faceted Approach to Knowledge Organisation and Retrieval in the Cultural Heritage Sector, *Digicult Thematic Issue - Resource Discovery Technologies for the Heritage Sector*, 6, pp. 28-33.

Turtle, H. and Croft, W. B. (1990). Inference Networks for Document Retrieval, in Proc. of the 13th Annual Int. ACM SIGIR Conference on Research and Development in Information Retrieval, Brussels, Belgium, pp. 1-24.

Vapnik, V. N. (1999). *The Nature of Statistical Learning Theory* (2nd Ed.), Springer-Verlag, Germany.

Verhoeff, J., Goffmann, W. and Belzer, J. (1961). Inefficiency of the Use of Boolean Functions for Information Retrieval Systems, *Communications of the ACM*, 4(12), pp. 557-558, 594.

Vintar, S., Buitelaar, P. and Volk, M. (2003). Semantic Relations in Concept-Based

Cross-Language Medical Information Retrieval, in Proc. of International Workshop on Adaptive Text Extraction and Mining, Catvat-Dubrovnik, Croatia, pp. 83-91.

Voss, A., Nakata, K., Juhnke, M. (1999). Concept Indexing, in Proc. of the International Conference on Supporting Group Work (SIGGROUP 1999), Phoenix, USA, pp. 1-10.

Vossen, P. (2003). Ontologies, in The Oxford Handbook of Computational Linguistics, Mitkov, R. (Ed.), Oxford University Press, Oxford, UK, pp 464-482.

Voutilainen, A. (2003). Part-of-Speech Tagging, in The Oxford Handbook of Computational Linguistics, Mitkov, R. (Ed.), Oxford University Press, Oxford, UK, pp. 219-232.

Wartick, S. (1992). Boolean Operations. in Information Retrieval: Data Structures and Algorithms, Frakes, W. B. and Baeza-Yates, R. (Eds.), Prentice Hall, USA, pp. 264-292.

Weisstein, E. W. (2005). <http://mathworld.wolfram.com/Hyperplane.html>, [Accessed on 16 Jan, 2006].

Wiig, K. M. (1995). Knowledge Management Foundations: Thinking About Thinking - How People and Organisations Create, Represent and Use Knowledge, Schema Press Ltd., USA.

Wilkinson, R. and Hingston, P. (1991). Using The Cosine Measure in a Neural

Network For Document Retrieval, in Proc. of ACM SIGIR Conference on Research and Development in Information Retrieval, Chicago, USA, pp. 202-210.

Witten, I. H., Moffat, A. and Bell, T. C. (1999). *Managing Gigabytes: Compressing and Indexing Documents and Images* (2nd Ed.), Morgan Kaufmann, USA.

Wong, S. K. M. and Yao, Y. Y. (1995). On Modelling Information Retrieval With Probabilistic Inference, in Proc. of ACM Transaction on Information Systems (TOIS), 13(1), pp. 38-68.

Wu, S. and Manber, U. (1992). Agrep - A Fast Approximate Pattern-matching Tool, in Proc. of USENIX Technical Conference, Francisco, USA, pp. 153-162.

Wu, S. and Manber, U. (1992). Fast Text Searching Allowing Errors, *Communications of the ACM*, 35(10), pp. 83-91.

Yang, C. C., Chen, H. and Hong, K. K., Visualisation Tools for Self-Organizing Maps, in Proc. of the 4th ACM Conference on Digital Libraries, Berkeley, USA, pp. 258-259.

Yarowsky, D. (1992). Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora, in Proc. of COLING'92, Nantes, France, pp. 454-460.

Yarowsky, D. (1993). One Sense Per Collocation, in Proc. of the 5th ARPA Human Language Technology Workshop, Princeton, USA, pp. 266-271.

Zadeh, L. A. (1965). Fuzzy Sets, *Information Control*, 8, pp. 338-353.

Zadeh, L. A. (1994). Fuzzy Logic, Neural Networks And Soft Computing, *Communication of ACM* 37 (3), pp. 77-84.

Zipf, G. K. (1949). *Human Behaviour and the Principle of Least-Effort*, Addison-Wesley, USA.

eTesting Labs Inc. (2000). *Google Web Search Evaluation*, eTesting Labs Inc, USA.

van Rijsbergen, C. J. (1979). *Information Retrieval (2nd Ed.)*, Butterworths, UK.

