Building a prototype for quality information retrieval from the World Wide Web

Milly Wei-Tsen Kc
University of Wollongong


This paper is posted at Research Online.
http://ro.uow.edu.au/theses/858
NOTE

This online version of the thesis may have different page formatting and pagination from the paper copy held in the University of Wollongong Library.

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
Building a Prototype for Quality Information Retrieval
from the World Wide Web

by
Milly Wei-Tsen Kc

A thesis submitted in partial fulfillment of the requirements for
the award of the degree Doctor of Philosophy
Faculty of Informatics
University of Wollongong

June 2009
This research project has been partially funded by Australian Research Council in the form of a Discovery Project Grant DP0452862 (2004-2006, extended to 2007), and a Linkage International Grant LX0454446 (2004, 2005).

This research was supervised by Dr. Markus Hagenbuchner and Prof. Ah Chung Tsoi
CERTIFICATION

I, Milly Wei-Tsen Kc, declare that this thesis, submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy, in the Faculty of Informatics, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

Milly Wei-Tsen Kc
22 June 2009
Given the phenomenal rate by which the World Wide Web is changing, retrieval methods and quality assurance have become bottleneck issues for many information retrieval services on the Internet, e.g. Web search engine designs. In this thesis, approaches that increase the efficiency of information retrieval methods, and provide quality assurance of information obtained from the Web, are developed through the implementation of a quality-focused information retrieval system.

A novel approach to the retrieval of quality information from the Internet is introduced. Implemented as a component of a vertical search application, this results in a focused crawler which is capable of retrieving quality information from the Internet. The three main contributions of this research are: (1) An effective and flexible crawling application that is well-suited for information retrieving tasks on the dynamic World Wide Web (WWW) is implemented. The resulting crawling application (crawler) is designed after having observed the dynamics of the web evolution through regular monitoring of the WWW; it also addresses the shortcomings of some existing crawlers, therefore presenting itself as a practical implementation. (2) A mechanism that converts human quality judgement through user surveys into an algorithm is developed, so that user perceptions of a set of criteria which may lead to determination of the quality content on the web pages concerned, can be applied to a large number of Web documents with minimal manual effort. This was obtained through a relatively large user survey which was conducted in a collaborative research work with Dr Shirlee-Ann Knight of Edith Cowan University. The survey was conducted to determine what criteria Web documents are perceived to meet to qualify as a quality document. This results in an aggregate numeric score for each web page between 0 and 1 respectively indicating that it does not meet any quality criteria, or that it meets all quality criteria perfectly. (3) This research proposes an approach to predict the quality of a web page before it is retrieved by a crawler. The approach allows its incorporation into a vertical search application which focuses on the retrieval of quality information. Experimental results on real world data show that the proposed approach is more effective than any other brute force approaches which have been published so far.

The proposed methods produce a numerical quality score for any text based Web document. This thesis will show that such a score can also be used as a web page ranking criterion for horizontal search engines. As part of this research project, this ranking scheme has been implemented and embedded into a working search engine. The observed user feedback confirms that search
results when ranked by quality scores satisfy user needs more satisfactorily than when ranked by other popular ranking schemes such as PageRank or relevancy ranking. It is also investigated whether the combination of quality score with existing ranking schemes can further enhance the user experience with search engines.
Contribution of this thesis

The contribution of this thesis is multi-fold. This is due to the fact that research on quality information retrieval mechanisms for the World Wide Web is only just evolving, and hence, datasets, domain knowledge, and suitable approaches had to be examined or realized. A successful investigation into quality retrieval methods required access to reliable testbeds. An analysis into existing testbeds revealed that they were incomplete or out-dated, and hence, were no longer reflecting WWW properties. As a result, we developed a distributed crawler which enabled us to retrieve accurate snapshots of a portion of the WWW at regular intervals. In addition, the work for this thesis required a good understanding of the behaviour of web page creation, evolution on the Internet. Existing literature analysed the properties of the WWW as was valid at the time of the examination. We examined the WWW properties on our snapshots in order to verify claims made by others, and in order to understand the WWW as it evolves over time, and detect their trends. The afore-mentioned tasks enabled us to address the quality information retrieval aspect of this thesis. As a result, the contributions of this thesis can be split into several parts as follows:

A.) Development of a scalable and accurate distributed crawler for the WWW: All crawlers known at the commencement of this project implement approximations or exhibit other limitations so as to maximize the throughput of the crawl, and hence, maximize the number of pages that can be retrieved within a given time frame. As a consequence, it is known that existing crawlers are not capable of obtaining accurate snapshots of the Internet. For the purpose of this research, it is essential to have access to an accurate and reliable testbed on which development and experiments can be based. As a consequence, we realized a distributed crawling concept which is designed to avoid such approximations, to reduce the network overhead, and runs on relatively inexpensive hardware. This allowed us to generate regular snapshots of portions of the Internet containing over 27 million web pages in each snapshot.

B.) The analysis of WWW properties, WWW dynamics, and trends: The Internet is continuously changing. It is known that the degree of change in the WWW follows an exponentially increasing curve. Hence, existing literature on WWW properties may no longer reliably reflect properties of the current Internet. This motivated us to verify statements made in the literature through an analysis of the snapshots of the WWW which we obtained at regular intervals. The analysis revealed up-to-date properties of the WWW, enabled us to understand its dynamics, and to detect its trends. The development of quality information retrieval methods benefits from such an analysis in that the awareness of actual changes in
the WWW is taken into account when addressing quality assessment criteria of web pages.

C.) A novel mechanism for predicting web page quality: The aim of any quality information retrieval system is to retrieve documents of high quality without having had prior access to these documents (i.e. to allow the evaluation of the quality of the document). It is thus required that a prediction mechanism to produce a recommendation regarding the order by which documents are presented from within a set of possible candidates. In other words, a mechanism is required which can estimate or predict the quality of a document before it is retrieved such that it becomes possible to decide on which of the possible documents should be retrieved next. This research deployed a machine learning approach to learn to predict document quality on the basis of knowledge about the document and its surroundings. More specifically, parent pages, the links, and the link structure are analysed for indications towards the quality of a target page.

D.) A novel ranking scheme for WWW documents: The method of producing a prediction for web page quality can be readily applied to assess the quality of pages in a web page repository. This associates a numeric value or vector to a document to indicate its quality. As a result, it becomes possible to sort the documents such that high quality documents are listed first whereas documents of lower quality are listed later. In practice, the ordering of web documents according to some criteria is known as web page ranking. Existing criteria are popularity which orders web documents by using link analysis techniques, and relevancy in which pages are ordered with respect to relevancy to a search criterion. This project produced a new web-page ranking criterion based on document quality. The process can be readily applied to realize Internet search engines which will return documents of high quality in response to a search query.

The following list of publications were a direct result of research performed in this thesis.


1 The list of publications is sorted by date of publication.


It should be noted that Wei-Tsen Milly Chiang changed her name to Milly Wei-Tsen Kc in 2006, and hence, there is a difference in name in the 2005 publication and subsequent publications.
Glossary

ANN  Artificial Neural Networks aim at emulating the behaviour of neurons or neural assemblies in the brain.

DAG  Directed acyclic graph.

DOAG  Directed ordered acyclic graph.

GraphSOM  A Self Organizing Map capable of processing many types of graphs.

HTML  This is a way to format a document using what is known as hypertext markup language, a special class of markup language for representing Internet documents.

INEX  This is an acronym for “INitiative for the Evaluation of XML Retrieval”, and refers to an international competition on XML structured document mining.

Internet  This refers to the large collection of online resources and services including the World Wide Web (WWW), email, file transfer and others.

Leaf node  is a node in a graph which has no outgoing links. This is sometimes called a frontier node.

Macro F1  A non-weighted performance measure. An average of $F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$

Micro F1  A weighted performance measure. Average F1 weighted by the number of documents in each class.

ML  Machine Learning.

MLP  Multilayer Perceptron is a neural network model based on artificial neurons that are arranged in layers.

MSE  Mean Squared Error.

Root node  is a node in a graph which has no incoming links.

SOM  Self Organizing Map, a neural network model where neurons are arranged on an n-dimensional grid, with $n = 2$ most commonly. This is used often used for the projection of high dimensional data to one with lower dimensions, with grid points being represented by neurons.

SOM-SD  Self Organizing Map for Structured Data. Similar to SOM but for the encoding of structured data.
CSOM-SD Contextual Self Organizing Map for Structured Data. This is a SOM-SD which includes the context of nodes to the learning process.

SSE Summed Squared Error.

TLD Top Level Domain, the end bit of a domain name. For example, “.de” is the TLD for the domain www.uni-ulm.de.

Tree A tree is a particular type of acyclic connected graphs where each node has at most one parent.

VQ Vector quantization.

Web A shortened form of World Wide Web, which generally refers to a system of documents accessible via the Internet.

Web document This refers to a document found on the World Wide Web which may be an HTML-formatted file, a plain text file or a binary file.

Web page This refers to a document which is formatted using the HTML convention.

WWW World Wide Web.
Notation

The following notations are used throughout this thesis. Scalars and constants are indicated by lowercase script letters e.g., \( c \). Parameters for dynamic processes are stated as lowercase Greek letters such as \( \alpha \). Vectors are denoted by lowercase bold letters, e.g., \( \mathbf{v} \). Sets and matrices are denoted by upper case letters, e.g., \( S \). Sometimes, in order to avoid confusion, we use uppercase bold letters e.g., \( \mathbf{M} \) to denote matrices. Calligraphic letters e.g., \( \mathcal{G} \) are used for representing graphs. Domains are indicated by bold calligraphic letters e.g., \( \mathcal{I} \). Lowercase script letters are used to access elements of a vector or matrix. As an example, in order to access the \( i \)-th element of a vector \( \mathbf{v} \) we use \( \mathbf{v}_i \). Letters when used in combination with brackets such as in \( f(x, y) \) denote functions. A few examples are given below:

\[
\begin{align*}
n &= |\mathbf{x}| \quad &\text{n is the dimension of vector } \mathbf{x} \\
\mathbf{x} &= (x_1, \ldots, x_n) \quad &\text{Vector } \mathbf{x} \text{ consisting of } n \text{ elements.} \\
F(\mathbf{x}) &= \quad &\text{A function taking a vector as argument.} \\
\mathbf{C} &= \mathbf{A}\mathbf{I} \quad &\text{\( C \) is the result of a matrix multiplication.} \\
W_{ij} &= \quad &\text{refers to the } ij\text{-th element of the matrix } \mathbf{W}. \\
S &= \{0,1,2\} \quad &\text{A set with three elements.} \\
\mathbf{m}_i &= \alpha \mathbf{m}_i \quad &\text{Recursive update of the } i\text{-th element of vector } \mathbf{m} \\
\alpha(t) &= \quad &\text{The parameter } \alpha \text{ depends on time } t.
\end{align*}
\]
Acknowledgment

I am grateful to Australian research Council (ARC) for the financial support provided in the form of an ARC Discovery Project Grant to Professor A. C. Tsoi which was subsequently transferred to Dr. M. Hagenbuchner, when Professor Tsoi became ineligible as he was employed by the ARC, which made this research project possible. My appreciation also goes to AC3 for providing high performance cluster computing facilities, staff at University of Sienna and University of Padova for providing much needed access to network and computing facilities, and staff at University of Wollongong for accommodating the special needs of this research. These all contributed to the successful execution of the experiments of this research.

Very importantly, I would like to thank my supervisors Dr. Markus Hagenbuchner and Prof. Ah Chung Tsoi for their support and guidance. I appreciate their friendship, valuable opinions and commitment for excellence. I would also like to thank the fantastic researchers I had the pleasure to collaborate with. They are Sweah Liang (Linus) Yong, Dr. Shirlee-Ann Knight, Prof. Franco Scarselli, Prof. Alessandro Sperduti and the many friendly researchers and staff at Edith Cowen University (WA, Australia) and University of Sienna (Italy) who made the collaboration a wonderful experience. My teachers in the past also need a mention for doing a great job passing the knowledge on, especially my honours thesis supervisor, Associate Prof. Carole Alcock, who lead me into the world of research with enthusiasm and provided much appreciated guidance.

On a more personal note, I would like to thank my husband Suresh and my son Ryan for being so understanding and helpful, not to mention always showering me with encouraging words. A big thank you also to my cool parents for providing me wonderful learning environments and the opportunities to explore anything and everything when I was younger. Finally, thanks to all who pitched in to help when I was away from home on research related trips. I am thankful for the many forms of support received during my research; they all contributed to the successful completion of this thesis.
# Contents

## 1 Introduction

1.1 Introduction to the topic area .................................................. 1
1.2 Research motivation .......................................................... 6
1.3 Aims and objectives ............................................................. 7
1.4 Research design ................................................................. 8
1.5 Research scope ................................................................. 9
1.6 Thesis outline ................................................................. 10

## 2 Background and motivation

2.1 Introduction ................................................................. 11
2.2 Domain knowledge ........................................................... 12
  2.2.1 Web size estimation ..................................................... 14
  2.2.2 The dynamics of the Web .............................................. 14
  2.2.3 Power-law distribution .................................................. 15
2.3 Search and retrieval on the World Wide Web .................................. 17
  2.3.1 Search engines ........................................................... 18
  2.3.2 Meta-search engines ..................................................... 21
  2.3.3 Web directories .......................................................... 22
2.4 Crawling models ............................................................. 24
  2.4.1 Basic crawling ........................................................... 24
  2.4.2 Focus crawling ........................................................... 25
  2.4.3 Parallel crawling ........................................................ 28
  2.4.4 Distributed crawling .................................................... 28
2.5 Scoring and ranking models .................................................. 33
  2.5.1 Link-based algorithms ................................................... 34
  2.5.2 Usage-based algorithms ............................................... 37
5 Document properties

5.1 Introduction ..................................... 127

5.2 Document structures .............................. 127
  5.2.1 Inter-document structure ................. 127
  5.2.2 Intra-document structure ................ 129

5.3 Document features ................................ 130
  5.3.1 Layout-based features ..................... 130
  5.3.2 Time-based features ...................... 131
  5.3.3 Usage-based features ................. 131
  5.3.4 Text-based features ...................... 132
  5.3.5 Semantic-based features .............. 133
  5.3.6 Miscellaneous features ........... 133
  5.3.7 Quality-based features ............. 134

5.4 Criteria for quality analysis .................. 135

5.5 Suitability of extracted feature for quality analysis ............... 139
  5.5.1 Accuracy .................................. 140
  5.5.2 Consistency ................................ 141
  5.5.3 Believability .............................. 142
  5.5.4 Understandability ......................... 143
  5.5.5 Completeness .............................. 144
  5.5.6 Concise ................................... 145
  5.5.7 Timeliness ................................. 145
  5.5.8 Security ................................... 145
  5.5.9 Objectivity ................................ 146
# 8 Related work

8.1 Introduction ...................................... 217
8.2 Changes in web search ............................... 217
  8.2.1 The provision of search services ...................... 218
  8.2.2 Extensions on searching functionalities ................. 220
8.3 Improvements in crawling efficiency ...................... 222
8.4 Web page ranking approaches ........................... 225
  8.4.1 Link analysis based ranking approach ................... 225
  8.4.2 Machine learning based quality evaluation ................. 229
8.5 Conclusion ....................................... 231

# 9 Discussion and conclusion

9.1 Introduction ...................................... 233
9.2 Findings and implications .............................. 234
  9.2.1 The need for automated quality information retrieval ........... 234
  9.2.2 The requirements of information retrieval experiments .......... 234
  9.2.3 The requirements of the quality evaluation process ............. 235
  9.2.4 The overall quality information retrieval system ............ 236
9.3 Contributions .................................... 236
9.4 Limitations ..................................... 238
9.5 Future work ..................................... 238

Bibliography

A Report of an informal interview with a librarian
List of Figures

2.1 Illustration of the various elements of an URL ................................. 13
2.2 A visualization of a minute portion of the Web using TouchGraph, showing the connectivity that can be found on the Web ................................. 15
2.3 The documents on the Web displays a bow-tie structure according to the power-law distribution ................................................................. 16
2.4 A diagram of the architecture of a general search engine .................... 18
2.5 A diagram of the architecture of a general Meta-search engine ............ 21
2.6 A possible file structure on a website for illustrating breadth-first and depth-first crawling order ................................................................. 24
2.7 An illustration of centralized topology for distributed crawling system ........ 29
2.8 An illustration of decentralized topologies for distributed crawling system .... 30

3.1 The vocabulary vector dot product score .......................................... 62
3.2 An example of a symbolic link structure ........................................... 65
3.3 The structure and components of the proposed distributed crawler .......... 70
3.4 Illustration of the crawling throughput achieved when increasing number of crawlers are adopted in parallel .................................................. 86
3.5 Illustration of the throughput over time during local and international crawl .... 87
3.6 The composition of various general TLDs in 1997 and 2004 respectively .... 97

4.1 The architecture of a simple Self-Organizing Map. Shown is a two-dimensional map with $8 \times 4$ neurons arranged on a hexagonal grid ........................................ 105
4.2 Visualization of the mapping for documents of classes 1-6 .................... 117
4.3 Visualization of the mapping for documents of classes 7-12 .................... 118
4.4 Visualization of the mapping for documents of classes 13-18 .................... 118
4.5 Performance comparison between SOM-SD and CSOM-SD when utilizing both structure and content information .......................................... 121
5.1 Illustration of domain based hierarchical structure .......................... 128

6.1 Illustration of Phase 1 in the quality assessment .......................... 163
6.2 Diagram of the machine learning task cycle ................................. 169
6.3 Illustration of the distribution of MSE rates .................................. 172
6.4 Part 1 of individual component error ........................................... 173
6.5 Part 2 of individual component error ........................................... 173
6.6 Scores of web pages in the order of crawling ................................ 179
6.7 Scores of web pages in crawling order with colour-coded sites .......... 180
6.8 Scores of web pages in crawling order with the boosting strategy ....... 181

7.1 Diagram showing the 5 major components in the crawling slave ......... 188
7.2 The retrieval rate of high quality web pages using different crawling methods on a dataset consisting of 30,869 pages ........................................ 205
7.3 The retrieval rate of high quality web pages in the early stage of crawling a large dataset of 26.6 million pages ........................................ 206
7.4 A screenshot of the search comparison platform ............................... 208
7.5 An illustration of the sliding window concept .................................. 213

8.1 The share of searches among various search service providers in July 2006 .... 218
8.2 The number of unique users for various types of search-related services provided by Google in December 2007 ................................. 222
8.3 Illustration of the correlation between pagerank and the proposed quality score . 227
8.4 The distribution of pagerank in the top 500,000 web pages of the 26.62 million pages testbed ..................................................... 229
List of Tables

2.1 Crawler performance as measured by practical features ........................................ 32
2.2 Quality criteria ordered according to their overall importance as recognized in literature ................................................................. 51
3.1 Number of papers which employ the WT10G and/or WT100G in experimental settings .................................................. 58
3.2 A comparison of the basic properties of the 2 TREC datasets ........................................ 59
3.3 A comparing of the statistical properties of the 2 TREC testbeds .................................. 60
3.4 Illustration of the effect of symbolic link on the link extraction process of crawling ................................................................. 66
3.5 The average throughput achieved by crawlers in various locations with various crawling approaches .................................................. 83
3.6 The crawling throughput achievable with 1 to 4 simultaneous processes ...................... 85
3.7 The basic statistics of the retrieved web snapshots ..................................................... 90
3.8 The statistical summary of the filtered testbeds developed from web snapshots ..................... 91
3.9 Accessibility of pages and domains in the WT10G collection. Valid as of September 2004 ................................................................. 94
3.10 The top 5 popular file extensions from all hyperlinks found in the web snapshots .............. 95
3.11 The top 5 popular unique file extensions per page in the web snapshots .............................................. 96
3.12 TDLs introduced after 2000 ........................................................................ 97
3.13 Properties of different types of domains ................................................................ 98
3.14 Rate of change in different types of domains ................................................................ 98
4.1 The topic area and types of journals in the INEX 2006 XML test-bed ................................ 113
4.2 The training parameters used for the structure only learning task .................................. 116
4.3 Training parameters used for the clustering of both structural and textual information ................................................................. 120
4.4 A comparison of test results for structure only clustering .............................................. 122
Chapter 1

Introduction

1.1 Introduction to the topic area

The World Wide Web is a relatively new and dynamic medium for information exchange and knowledge sharing, made possible through the introduction of the Internet. Even though the World Wide Web is currently the most commonly used medium for sharing information and knowledge, it has only been available for public utilization on the Internet since 1991[49]. The World Wide Web defines a high level communication protocol for computers connected via the Internet.

The Internet provides the physical possibility for computers to communicate through a line connection\(^1\). The Internet was developed in the 1970s so as to allow the communication between mainframe computers over some distance. The emerging availability of end user computers, and their rapidly increasing popularity caused the Internet to become more wide spread. The Internet provides a physical line and a low level protocol (TCP/IP) for information exchange. The definition of a higher level File Transfer Protocol (FTP) in the 1970s provided more user friendly means for information exchange[49].

FTP is used for transferring files from one computer to another, this is typically done by logging into an FTP server, and then either perform a file download or a file upload. Since FTP is only used for file transfers between computers, and the destination computer may not belong to the user, the task would be difficult and time consuming if the location and name of the desired file is not known. A service called Archie was developed to address this issue by providing keyword searching of files. However, Archie was a rather primitive search service, it only provides string matching between search term provided by the user and substrings of file

\(^1\)The definition of a line connection is being softened through the introduction of the wireless communication medium. Hence, a communication line is said to be the physical medium which allows point to point communication.
names. The content of the file was not analyzed. This provided little help, as not all files used names that suggest its content. User friendly graphical user interfaces for FTP clients were later also developed but these did nothing to help support the finding of relevant information on the Internet.

FTP is limited to the exchange of files, and is unsuitable for the exchange of messages, or the delivery of embedded content. This former was addressed with the introduction of the Internet Relay Chat (IRC) in 1988 which allowed the exchange of information and messages between members of a group (organized in channels). The later was addressed with the introduction of Gopher in 1990. Gopher allowed server based text files to be hierarchically organised and easily viewed by end users. Gopher is the precursor to the World Wide Web which had introduced a browsing and retrieval system that includes a user interface with menus, and this addressed some of the shortcomings of FTP. A search facility called Veronica was developed, and its benefits were similar to those provided by Archie for FTP. However, Gopher does not have multimedia support and it performed tasks in a linear manner, therefore became useless and eventually extinct, when the size of the World Wide Web grew rapidly. Hence, the World Wide Web is the result of developments towards a user friendly medium for information exchange in a possibly large system connected via the Internet. It defines a high level protocol for communications over the Internet[49].

After the World Wide Web was available for use by the public, users could establish an Internet connection, and through that, publish information on the World Wide Web, or access information published by other users via a browser. The World Wide Web has made the sharing of information in ways that were not possible before. The fact that the information published on the World Wide Web is openly available for access any time of the day and from any location with access to Internet means that information sharing is not restricted by location, distance or time-zone. Therefore, theoretically, any person is able to publish any information on the World Wide Web, and any information on the World Wide Web can be accessed by anyone. This made the World Wide Web hugely popular with an exponentially increasing amount of information being added or modified continuously. The result is that the World Wide Web is a medium under development. Processes that are based on the Internet often require regular updates in their algorithms to account for the changes in the World Wide Web. Example of such processes are Web search engines, web directories, digital libraries, and many more.

The growth of the World Wide Web makes it an increasingly rich source of information, but with an increasing amount of information on the World Wide Web, users were becoming overwhelmed by the amount of data they have to deal with, and it was always a challenge to discover
the information of interest. To solve this issue, a mechanism was implemented to allow users to search for the information of interest from the vast amount of data in the World Wide Web. This search mechanism was in the form of search engines, web directories or digital libraries, which are generally categorized as information retrieval systems. Such systems are often designed to deal with very large amount of information (in the order of terra-bytes), and serve a possibly large number of users. As a consequence, information retrieval systems are often realized as a distributed system which parallels the underlying tasks.

Information retrieval refers to the process of searching for document, searching for information within a document or even for information about a document, and the systems aforementioned are called information retrieval system because they perform such a function to enable the provision of their service on the World Wide Web. Information retrieval systems may appear to retrieve information directly from the World Wide Web, but they actually retrieve the information the user may be seeking from a large data repository which contain subsets of the World Wide Web instead, and then display the results that match the user’s information seeking criteria to an interface.

There are two types of information retrieval systems: vertical and horizontal systems. These can be described as follows:

**Horizontal information retrieval systems:** Such systems retrieve information about the World Wide Web through *crawling*, the accessing and downloading of as many as possible documents which can be found on the Web. This can be a time consuming task due to limitations in the bandwidth that is available to the system. Once loaded, the retrieved information is indexed and locally stored (in a possibly compressed form). Thus, horizontal information retrieval systems create a snapshot of the World Wide Web where the coverage is limited by the time needed to create the snapshot. This allows for a much accelerated access to information which is available on the Web, but requires regular updates of the cached snapshot. The functionality (i.e. search service) is then executed on the locally cached snapshot rather than on the live Web.

**Vertical information retrieval systems:** Such systems are a specialization of horizontal information retrieval systems to a particular user need. For example, to serve users which are exclusively interested in astronomy or cooking. In such circumstances, it does not make sense to indiscriminately retrieve large snapshots of the World Wide Web (at a cost). Vertical information retrieval systems realize a concept of *focused crawling* which aim at retrieving (accessing and downloading) as accurately as possible only those documents on the Web which meet a certain criteria (i.e. address the topic astronomy or cooking).
The most important difference between these two types of systems is that the former applies its functionality to locally available data, and hence, accesses documents whose content is known. In contrast, vertical information retrieval systems require the finding of documents which may not yet be in a local cache, and hence, require mechanisms that can predict the content of a document before its retrieval. This approaches proposed in this thesis will consider both types of mechanisms in information retrieval systems.

It should be noted that although the information retrieval systems aim to provide results which satisfy the information needs of the user, that is not always the case. There are a number of reasons for this:

1. It is not possible for any information retrieval system to have knowledge of a significant portion of the extremely dynamic and unimaginably large World Wide Web. This is because of limitations in the crawling technology, in the achievable bandwidth and in the data processing or indexing tools, which cannot scale to a similar magnitude as the World Wide Web. This is closely related to the synchronisation paradigm in distributed systems which introduce inconsistency measures to indicate the deviation between replicas. In other words, in large scale distributed systems it is often not possible to maintain exact copies amongst its replicas. The World Wide Web can be seen as a very large scale distributed system, and information retrieval as a tool to attempt synchronization of Web content with a local data store. The World Wide Web domain further amplifies the issue with the growth of population with access to the Internet, and the common practice of having computers constantly connected to the Internet which promotes the rapid growth of the World Wide Web. Moreover, there is an increasingly popular possibility to have automated programs that provide (dynamic) web content which cause the size of the Web to be virtually infinite though only a relatively small part of it is unique, and contains useful information.

2. Information retrieval systems struggle to provide information to meet user’s information needs is because the open and nonrestrictive nature of the World Wide Web is unlike the traditional paper-based information, as such, there is no standard or quality control over the published information. This means that anyone can publish any data on the World Wide Web with no obligation to ensure the correctness or the significance of the content. Publishing on the World Wide Web was an activity previously carried out by an elite group

---

2A simple script which generates a random number from $R$, or reports the current time in response to an access via the World Wide Web would produce an infinite amount of information (since every access would produce a different response) but would contain information which is of little use to the user of the World wide Web.
of people with sufficient technological knowledge as an academic tool, as those people would need to know how to code in HyperText Markup Language (HTML), how to host a website and how to transfer the files to the hosting machine. However, the introduction of the first browser called Mosaic and the web publishing tools that were developed afterwards, made publishing on the World Wide Web much easier that even people with no technological background at all, such as teenage children, can publish[49]. Therefore it is highly possible for search results to contain information that is of little value and low quality.

3. There is currently no algorithm that is able to identify the most appropriate web pages, to meet the information needs of the users. There are search engines built on automated applications to crawl the World Wide Web, to identify relevant documents, and to rank the documents before displaying the result set to the users. There are also systems that took a different approach; instead of collecting and searching through a mass of web pages, they focus on selecting and storing web pages that human information experts agree to be of an acceptable quality. Although this approach ensures a result set of high quality documents, the number of documents the system has access to, is very limited. As a result, some searches would return no, or very little number of matching documents, and many important documents are not included in the result set because the system is simply not aware of them. All these are reasons which make quality information retrieval from the World Wide Web challenging.

In fact, the analysis and development of information retrieval systems have been a focus of research for some years [59, 130, 116], with the aim of improving the existing information retrieval systems, so that they are able to provide results that better satisfy the information needs of the user. Such a task involves investigation and study in the following areas (incomplete list):

- **Back-end crawling process** - Required to traverse the World Wide Web for the retrieval of web pages into a data repository

- **Back-end indexing process** - Required to label the retrieved web pages and to allow the search and identification of matching web pages in an efficient manner.

- **Distributed database design** - Required to parallel accesses to the large repository of a locally stored snapshot of the Web.

- **Document ranking** - Required to define an order on documents which match a given query.
• Front-end user search interface - Required to allow the input of search query terms and the display of results in some ranked order

There are many more issues that require consideration in designing information retrieval systems due to issues concerning security, transparency, scalability, consistency, accuracy, fault tolerance, etc. This exemplifies the level of complexity involved in the design of information retrieval systems.

This thesis focuses on investigating ways to improve the quality of information retrieval results. This can be carried out through the development of a prototype for quality-based information retrieval. In this research, a novel approach is proposed to combine efficient information retrieval with an automated quality evaluation mechanism that is based on empirical findings.

1.2 Research motivation

Information retrieval has been an area of extensive study, especially in recent years [33, 75], due to the size and the dynamics of the Web, coupled with the increased use of search services to locate information. Although search systems are widely used for the identification of desired information, this is not achieved in practice. Large-scale information retrieval systems have access to a large amount of data, but do not offer quality assurance, and existing high-quality information retrieval systems do not have sufficient coverage of the information available on the World Wide Web, to be rendered useful. This is because the quality evaluation process is currently manually achieved due to its heavy dependence on human cognitive decision, which is challenging to interpret by computing facilities, and therefore is challenging to apply to the vast and continuously growing Web.

This thesis addresses such shortcomings of current information retrieval systems, and aims to close the gap between theoretically based quality models and the application of quality information retrieval on a large quantity of data. The result is a prototype of an information retrieval system that is able to retrieve information efficiently and effectively, while incorporating an automated quality evaluation mechanism. The proposed system is able to deal with the large amount of data on the World Wide Web, as well as provide an enhanced level of quality assurance to search results.

The relevancy of this research is amplified by recently published works made by others aiming at quality information retrieval from the Internet [3, 67, 103]. This thesis proposes an approach which is proven to be superior to those competing approaches. This will be shown in chapter 2 and chapter 8 of this thesis.
1.3 Aims and objectives

This research aims at quality information retrieval from the World Wide Web with applications to search engine design and digital libraries. This is to be achieved by completing the following tasks, which focus on information retrieval and quality evaluation aspects.

Step 1: Investigate existing information retrieval systems.

Step 2: Evaluate the usefulness of existing crawling strategies.

Step 3: Identify or develop an efficient and effective crawling application.

Step 4: Verify the performance of the crawling application.

Step 5: Select or construct an appropriate information retrieval test-bed for experiments.

Step 6: Examine quality evaluation criteria through reviewing the literature.

Step 7: Analyze approaches that could interpret the human cognitive process of evaluating quality into machine understandable algorithm

Step 8: Identify or develop a practical quality evaluation framework.

Step 9: Incorporate the quality evaluation framework into the crawling application.

Step 10: Observe the quality evaluation feature to ensure execution match design expectations.

Step 11: Verify the overall system performance.

Step 12: Compare the performance of the proposed information retrieval system to other existing systems.

The result of this work are proposed solutions to:

A.) the realization of vertical information retrieval system with a focus on retrieving quality information. In other words, the thesis proposes mechanisms which allow the implementation of a focused crawler which is capable of predominantly retrieving documents from the internet which are of highest quality.

B.) the realization of horizontal information retrieval systems which return quality information in response to a query. This is achieved by defining a metric which encompasses the quality of documents in a local snapshot, and to impose an order on the documents so that these can be sorted from “highest quality” to “lowest quality”.
Note that in this thesis we refer to the term *quality* as the quality of a document as is perceived by the user. In other words, a users’ impression on the quality of a document when exposed to the document (i.e. the user sees the document in a web browser) forms the basis of the work presented in this thesis. This is made possible through an award winning user survey on quality perception of Web documents, and for which we propose algorithmic interpretations so as to allow for an automated evaluation of document quality.

The final outcome of this thesis is a working prototype system which encompasses quality assessment methods as are proposed in this thesis.

### 1.4 Research design

There are 2 major components in this research: information retrieval and quality evaluation. The proposed design for these two aspects will be described individually:

**Information retrieval:** The design of this research from the information retrieval aspect will require a crawling application which would be practical for retrieving information from the World Wide Web. This could be achieved through minor modification of an existing crawler, or through the design and development of a new crawler. Some analysis on the properties of the current World Wide web will be required in order to assess the suitability of crawlers for this task. A testbed would also be required so that experiments on web pages can be carried out in a controlled environment, which allows the reproduction and verification of experimental results.

**Quality assessment:** The design of this research from the quality evaluation aspect will require an automated mechanism for the evaluation and scoring of web pages, so that a score could be produced for each web page which indicates its quality. This aspect of the research is built on the findings of information quality from our research partner in Edith Cowen University, in Western Australia, whose work [75] received the 2008 Ballou & Pazer DQ/IQ Best Dissertation Award from Massachusetts Institute of Technology (MIT). With such a strong quality framework as the foundation, this research investigates mechanisms to transform the defined quality which is from a human’s cognitive interpretation, to machine implementable algorithms, and incorporates the most appropriate mechanism into an information retrieval system. In order to achieve this, machine learning methods are proposed. Machine learning methods are expected to be able to learn the relationships of pages and to identify patterns that could differentiate the web pages according to their level of quality, which will allow the web pages to be ranked according to their level of quality. For this
task, a testbed will also be required in order to assess the performance of the proposed quality evaluation algorithm.

Once both aspects of the research are fully implemented and tested on appropriate testbeds, they will be integrated and tested again, to ensure that the prototype is able to perform quality information retrieval in an automated, relatively efficient, and effective manner, in the World Wide Web.

1.5 Research scope

This research project aims to develop a prototype for quality-focused information retrieval. It should be noted that quality information retrieval refers to the retrieval of quality information, and not the quality of the information retrieval system. Furthermore, the definition of quality or what constitutes quality will not be a topic of this research, instead, it has been the topic of investigation for our research partner in Edith Cowen University, in Western Australia. This research project will utilize the findings from our research partner as the basis for the quality aspects of this research.

The scope of this research in the information retrieval aspect should also be defined. The information retrieval as carried out for this research will be on the a sufficiently large subset of the publicly accessible and indexable portion of the World Wide Web, which means the private or hidden web and portions of the World Wide Web which require authorized access are outside the scope of this research.

Lastly, the development of the prototype focuses on improving the backend applications, more specifically, the crawling and ranking components of existing information retrieval systems. Research into the indexing process and a search user-interface will not be provided in this project, as they are not within the scope of this research. It should be noted that this research project aims to develop a prototype of a search system, instead of a publicly accessible functional system, as utilizing the proposed crawling and quality ranking components alone will not provide a fully functional system. The crawling and ranking components proposed in this research project are developed with the purpose of being able to be adopted by existing information retrieval systems to allow for an improvement in the quality of their search results.

If a new and fully functional system is desired, an indexing application and a user interface will be required in addition to the crawling and quality ranking components proposed and implemented in this research project. The indexing application will be required to index the data crawled by the proposed crawler, and to label the quality score correspondingly for ranking pur-
poses. In addition, a user interface will also be needed to enable users to specify their query terms, to search through the index, and to display the results according to the quality rank in the label.

1.6 Thesis outline

The thesis is organized as follows:

Chapter 1 provides an overview of the research project, with emphasis on the research significance and its goals.

Chapter 2 gives background knowledge for this research topic through discussions of literature in the relevant topic area.

Chapter 3 explores collections of data currently available as experimental testbeds. The suitability of the collections as testbeds for this research is examined, with the development of new testbeds as a possibility as well.

Chapter 4 investigates the possible natural grouping of web documents through an unsupervised machine learning approach, and whether the grouping can provide indication of quality. The factors that influence the grouping are also explored.

Chapter 5 examines the various features that can be extracted from analyzing web documents, and the possibility of their utilization to assist in determining the quality of web documents.

Chapter 6 discusses the approaches in which quality is defined and evaluated.

Chapter 7 details the implementation of the quality information retrieval system.

Chapter 8 describes current developments in relevant areas of research.

Chapter 9 summarizes the findings and provides an indication for future research directions.
Chapter 2

Background and motivation

2.1 Introduction

The Internet is a fairly new medium for information exchange, its core idea of universal networking began in the late 1950s. Internet provides new approaches to communication, and Internet-based resources and services such as electronic mail, file transfer, online gaming and many more developed soon after. Among the many resources and services, the World Wide Web, invented by Tim Berners-Lee in 1989, undoubtedly contributes most significantly to global information and knowledge sharing.

On the World Wide Web, documents are interconnected via hyperlinks, and are openly available in most cases. Anyone with an Internet access can access or create documents on the World Wide Web. The rapid advances in technology in the recent decades provided more people with access to the Internet, and that in turn, resulted in the request for and the creation of more documents on the World Wide Web. Although the number of documents in the World Wide Web has increased, the user’s knowledge of the web pages was very limited, as there were no means of discovering web pages of interest unless the user has prior knowledge of them. The development of a web directory helped to solve this issue by collecting and organizing web pages into structures of categories; this is similar to providing human with prior knowledge of a larger number of documents, then extract the needed information based on this prior knowledge. Search engines were later implemented to assist users in locating web pages in a even more efficient manner, but they are also based on the same idea. Therefore, the process of extracting some information from the web, as carried out by web directories and search engines, is called information retrieval.

As the number of documents in the World Wide Web increases, the variety in the document content also increased. Some of the documents are of an acceptable quality, but since there is no restriction or enforceable rule on the contents posted in the World Wide Web, there are also
some documents with poor quality. Many of the systems that carry out information retrieval do not differentiate the good from the bad, and that has an impact on the ability for users to discover good source of information on the current World Wide Web. The goal of this research project is to address this issue, and propose an information retrieval system which makes a distinction between documents of acceptable quality and documents of poor quality on the World Wide Web.

Since the overall goal of this project is to develop a prototype for an information retrieval system that delivers high quality results on the World Wide Web, the domain knowledge about the World Wide Web will need to be investigated first. Then it is essential to examine the literature for search systems in general as well as the work that have been carried out in each of the building processes. Some existing quality frameworks will be investigated, and finally, some background information on machine learning approaches to quality estimation will be provided.

### 2.2 Domain knowledge

It is commonly known that the World Wide Web is enormous in size and dynamic in nature. Some literature attempt to estimate the size of the Web such as in [51, 38, 94], or observe the changes in the Web over a period of time such as [100, 97], all of which revealed interesting properties of the Web. The nature of this research is very much web-based, thus it would also be beneficial to observe and understand the characteristics of the Web, as it would assist in designing a quality information retrieval system that is suitable for utilization on the World Wide Web.

First of all, some commonly used terminologies and the overall structure of the World Wide Web should be defined to allow a clear understanding of this thesis. The World Wide Web is defined as “a system of interlinked hypertext documents accessed via the Internet” [135], and is commonly shortened to “the Web”. The documents on the Web are interlinked through hyperlinks, where throughout the thesis, Web will refer to the publicly accessible portion of the entire Web, which excludes the “hidden” web, and the web pages that require authorized access. Viewing a document on the Web is usually done by either following the hyperlink to that document, or by typing the Uniform Resource Locator (URL) of the page into a web browser.

URL helps to identify a document on the Web similar to the way an address helps to locate a property, therefore is sometimes informally named the web address. URL follows a structured naming scheme which usually has three components as in the sample shown in Figure 2.1. The scheme name is assumed to be http:// by default and is case-insensitive. HTTP refers to the “Hyper-text Transport Protocol” referring to the high level communication protocol commonly used in the World Wide Web. The host name is mandatory, and can be represented by a valid
domain name or an IP address. Note that host names are case-insensitive. A colon and a port number at the end of the host name is optional, and if not supplied as is the case in the example of Figure 2.1, the default port number of 80 will be used. The final portion of the URL called path name is optional; however, if the URL does not end with a specific filename as is the case in the example, file names such as index.html or default.html will be searched and displayed by default. The path name follows the naming schema of the file system on the host computer. Hence, on a UNIX host forward slashes ‘/’ are used to separate directory names whereas on a Windows host backward slashes are common.

![URL Diagram](http://www.uow.edu.au/student/)

Figure 2.1: Illustration of the various elements of an URL

When the domain name is used in the URL, the domain name is resolved into an IP address using the global and distributed Domain Name System (DNS). This is because although an IP address cannot be easily remembered by Web users, it is a necessary piece of information in order to contact and send data packets to the web server on which the document represented by the URL is located. All the domain names that resolve to the same IP address are said to be located on the same web server. Although some of the following terminologies are defined differently in various sources, they will refer to the corresponding definition provided here when they occur within the thesis.

- **Web document** - Data on the Web, includes binary and ASCII data.

- **Web page** - An ASCII text based document, usually in HTML or other similar syntax which can be interpreted by a web browser.

- **Domain** - Defined by unique domain names. Web documents with the same domain name in the URL are said to belong to the same domain.

- **Site** - Defined by a unique IP address. Since several domain names may resolve to the same IP address, and hence, this usually refers to the collection of web documents that are hosted on the same web server. Due to the increasing use of virtual hosting where an IP address can be shared among machines, or a machine can host several IP addresses, the boundary of site has become somewhat blurred.
• Top Level Domain (TLD)- The last segment(s) of the domain name. There are two major types of TLDs, the generic TLD (gTLD) and the country-code TLD (ccTLD). In the example in Figure 2.1, “edu” is the gTLD and “au” is the ccTLD.

The final three terminologies may be used to make a distinction in the granularity of grouping; for example, web documents can be grouped by domain, multiple domains can belong to the same site, and so on. The analysis of Web statistics is usually based on one or more of these groupings, as it is possible to maintain the exact number of domains, sites or TLDs, but it is not possible to keep record of the exact number of web documents on the Web. Only an estimation of the total number of web documents can be offered.

2.2.1 Web size estimation

There are many attempts to estimate the size of the Web [51, 38, 94]. The reason that the estimation of the size of the Web is so challenging is due to its sheer size, and its heavily connected property, which causes the discovery of new web pages time-consuming when a large portion is already known. The connectivity of the web also does not guarantee that all web pages are connected. In fact, it is quite common to find groups of web pages that do not connect to any other groups of pages. One approach of representing the connectivity of the Web and the isolated groups of web page is through a visualization software called TouchGraph [129] as seen in Figure 2.2. Because of the common formation of isolated groups (islands) of web pages, the estimation of the Web size is even more challenging. This is because there is no connection to these web pages and the other pages in these islands, therefore, they will only be known if a web page within the group is manually identified.

One of the most recent estimation was based on analyzing pages indexed by various search engines, which indicated that there are more than 11.5 billion indexed web pages in 2005 [51], and this grew to at least 17 billion by February 2007 [38]. An estimation of the publicly accessible portion of the Web claims that there were 433.19 million domains and 108.81 million sites in February 2007 [94].

2.2.2 The dynamics of the Web

Understanding the approximate size of the Web provides an awareness of the amount of process required by applications designed for the web; however, a more useful approach would be to observe the type and amount of changes, which may allow prediction of possible future changes on the Web.
The Online Computer Library Center conducted a routine analysis of the Web from 1998 to 2002 [100], which revealed some significant trends in the way that the Web is changing. One of the observations is the decrease in the proportion of the publicly accessible Web and the gradual increase in the proportion of the private web (intranet), where a private web is defined as websites where the content is intended for restricted audience, controlled in the form of authorization or membership requirement [100].

Another observation is the rate of website inaccessibility, consistently at approximately 50% per year. The observation on the evolution of the Web is extended to web pages, where the inaccessibility is at the extreme rate of 80% per year [97].

2.2.3 Power-law distribution

One of the major characteristic of the connectivities of documents on the Web is the power-law distribution. Most views of the Web assumed that all documents are interlinked, however,
observation made by [21, 117] showed that the distribution of web pages according to the link structure follow the power-law. Power-law states that “the probability that a node has (in- or out-) degree $d$ is proportional to $\frac{1}{x^d}$ for some $d > 1$” [117]. The link structure of most web crawls of various sizes also exhibit the Power law distribution, resulting in a bow-tie structure as illustrated in Figure 2.3, where web pages are classified into one of the following, listed in the order of the proportion of pages in each component.

- **Strongly Connected Component (SCC)**
  - Contains web pages that have links both to and from other pages in SCC

- **IN or OUT**
  - Contains web pages that have links either to or from pages in SCC
  - a balanced bow tie has similar number of pages in IN and OUT

- **Tendrils**
  - Contains web pages that have links either to or from pages not in SCC

- **Disconnected**
  - Contains web pages that do not have links

The power-law distribution describes the relationships of web pages on the Web through link structure, and is evident in web crawls that have not been processed or filtered [21, 117]. The above mentioned characteristics of the Web are considered domain knowledge, as these
observations of the Web property were publicly available before, or at the time when the current research began. It will be shown in Chapter 3 that during the course of the research, some of these domain knowledge were challenged for their validity in the current Web, and properties of the Web which were not previously known were observed and recorded.

2.3 Search and retrieval on the World Wide Web

Information retrieval systems have become essential tools for locating information on the gigantic Web, and they have evolved greatly in response to its increasing popularity and the rapid growth of the Web. A typical information retrieval process of information from the Web works in the following manner [19]:

1. A crawler will follow the hyperlinks contained on web pages. An initial set of web pages that a retrieval process begins with, is referred to as seed pages. The web pages pointed to by hyperlinks are prioritized by some pre-defined order for retrieval. The ordering could be breadth-first, depth-first or other ordering pre-defined algorithm. The crawler is expected to retrieve web pages in such an order.

2. The retrieved web pages are said to be crawled. Those pages are downloaded into an information storage system locally. We will call this local information storage an information vault. The crawled pages will accumulate in the information vault.

3. Information contained in the web pages in the information vault will be indexed. To enable retrieval of information, a reverse index is built which can be used to locate web pages which contained particular information.

4. The web pages contained in the information vault can be ranked using certain criterion. The most often used criterion is on how popular the web page is, judging by the number of pages which are linked to it. An example of such ranking algorithm is the well-known PageRank algorithm [19].

5. A user interface will return a list of web pages ranked according to, say, their popularity in response to a user query.

6. The user can click on any of the links, and can then be brought to the web page on the Web. Note that there may be some differences between the retrieved web page stored in the information vault and the actual live web page on the Web, especially for those web pages which have a tendency to change their contents frequently or if the consequent
There are extensive developments in recent years, but information retrieval systems each use a different approach in the design of their crawling application and ranking algorithm, in order to maintain an effective search system and ensure differentiation from other systems. The exact algorithms as is used in commonly-used information retrieval systems at current cannot be discussed in detail, as they are not disclosed for competitive reasons as well as to avoid misuse. Therefore, this chapter examines only the core crawling and ranking mechanisms that current information retrieval systems are based on, from the information that are publicly available or with sufficient documentation. The following are the three major categories of search systems that information seekers could utilize to locate information on the web, and they are search engines, meta-search engines and web directories.

2.3.1 Search engines

Search engines are information retrieval systems that maintain an indexing system built by crawlers that traverse the web to collect data, and allow users to search through the index with their query terms. The general architecture of a search engine type of information retrieval system is as illustrated in Figure 2.4. Notice that the building of an index through a crawling process and the sorting of query results through a scoring or ranking process are quite invisible to the users. Users only interact with an user interface, which contains a text box for entering their query terms, and then displays a list of search results. Therefore minimal web-based knowledge from information seekers is required.

The earliest generation of search engines are horizontal search engines that collect a repository of web pages in the information vault, then apply query matching technique where the
content of a web page was analyzed according to the frequency and location of word occurrences [24]. For example, web pages with a match in the title would lead the result list, followed by web pages where the search term (query) occurred several times in the content area [96]. However, the query matching techniques matched literal search terms, which means that frequently occurring words such as “the”, “to”, “is”, among others, were not filtered out. Also, when analyzing the web page contents or the search terms, the following were not considered.

- Syntax - the position of a word within a sentence structure were not considered
- Synonyms - synonyms of search terms were not considered in attempt to expand the query
- Polysemy - words with multiple meanings were not differentiated

Instead, the first generation of search engines considered the size of their index to be an important indicator of the search engine supremacy, as early studies into Web search engines reported that more than 30% of user queries generated a zero hit result [134]. Although this trend was later reversed with users reporting their queries produced far too many results, a growing index size was and is still being maintained by search engines. In the pursuit for the largest index, Alta Vista was the dominating search engine with the largest index in 1996, then Northern light was introduced with an even larger index, therefore became the dominating search engine in 1997. Northern light remained the largest until FAST was introduced in 1999 with 200 million index web pages [50], at which time, Northern Light was reported to index 16% of the Web [9].

With the increasing index size when the pursuit of the largest index began, the issue of zero hit result was solved, but the new issue of too many query results became a challenge for search engines. There were attempts to solve the problem by changing the default Boolean logic of ”OR” between query keywords to ”AND”, which decreased the number of results returned. However, the usefulness of the first generation search engines was visibly deteriorating as the size of the Web became too large to index a significant portion and the systems were easily abused, leaving good web pages buried amidst a mass of query results.

The contest on index size continued into the next generation of information retrieval systems, until Google announced that one billion web pages were indexed by its system, and claiming that due to the link and anchor analysis incorporated in the system, the number of web pages accessible by the search system is much more than one billion [96]. The size of Google’s index continued to grow with reports of 3.3 billion pages by August 2003 [123] and 4.2 billion pages in February 2004 [9]. During this time of dominance by Google, the search engine with the second largest index - AlltheWeb, only indexed approximately 3.1 billion pages in February 2004 [9]. Even though a large number of web pages can be indexed by current search engines, the growth
of the Web is estimated to grow exponentially as well, resulting in less than 10% of the current web being indexed, largely out-pacing the growth of indices.

The next generation of search engines were introduced with basic improvements over the first generation search capabilities, including removing frequently used words from query and using thesaurus to expand query terms [15, 87]. An example of second generation search engines is Google, which in addition to the basic improvements, features a ranking algorithm (Pagerank) that sorts the query results according to the number and types of web pages linking to a particular web page [24]. This ensures that the more recognized web pages are on top of the result list. The unique Pagerank algorithm coupled with Google’s simple search interface, allowed Google to become popular among users in a very short time. Although Google’s Pagerank is mostly based on popularity rather than quality [32], users were beginning to gain confidence in search engine, finding it an effective tool to locate information on the web.

Developments in other search engines have been carried out constantly in the mean time, some search engines were introduced with novel features, others upgraded themselves from the first generation to second generation. The developments were in various directions. The following examples show some development directions taken in the second generation of search engines.

- Search within specific fields

  AltaVista incorporated search in specific fields, such as “title:words” for searching within the title and “url:.au” for search within the URL [96]. This feature assists users who are seeking a more specific piece of information.

- Natural language search

  Ask Jeeves and Electric Monk developed natural language search, so that users enter query in the same way that they would ask a question [50]. This search method helps the search engine to identify the type of information users are seeking.

- Query results biased by past user behaviour

  Direct Hit and Hot Bot both offer query results based on the web sites user chose to visit [50]. If a web page displayed as the top query result was not visited, its rank would decrease, whereas if a web page with much lower rank, for instance a page listed on the eleventh page, was visited by users, its rank would increase significantly.

Apart from the modifications to search engines’ overall concept and front-end functionality, the back-end applications, such as crawlers, are being explored for possibilities of improving
efficiency, and of dealing with increasing amount of data by extending existing crawling models. The scoring or ranking mechanisms are also being examined for better indication of quality and ways to avoid abuse. These investigations into better crawling strategies and ranking mechanisms are described in detail in Sections 2.4 and 2.5 respectively.

2.3.2 Meta-search engines

Meta-search engines are a relatively new concept that utilize existing search systems that are publicly available, and therefore do not require the back-end development of information retrieving crawler or an index [91]. The concept is to utilize a number of search engines so that a query is sent to the selected group of search engines, and the results combined and displayed in one page to the user as illustrated in Figure 2.5.

Clyde [34] showed that although the number of web pages indexed by individual search engines are insignificant portions of the World Wide Web; however, the overlap in the indexed data among the various search engines is quite small. Therefore, Meta-search engines take advantage of this fact and combine a number of search engines in order to access the amount of indexed resources that is not achievable by a single search engine. The disadvantage of meta-search engines though, is that the performance of the search systems is largely dependent on the search engines employed as the source, but the scoring or ranking algorithms of those search engines cannot be utilized by Meta-search engines [141].

Older developments of Meta-search engine list results from various search engines separately and make no attempt at merging the results. Therefore it was common to have a web page listed in the query result multiple times. This problem has been addressed in newer Meta-search engines. Some Meta-search engines even attempt to differentiate themselves by emphasizing the development and the use of their own algorithm to rank and sort the query results [91, 141]. Examples of Meta-search engines include the following.

- Dogpile - searches through eleven search engines, but query results may contain duplication.
• Mamma - searches through eight search engines and the query results are sorted using the search engine’s own algorithm with duplication removal feature.

• InfoGrid - searches through a staggering sixteen search engines. Allows the order of the query result to be manipulated by user, with choices of sorting according to the relevance, title or source search engine. Also incorporates a search monitoring capability.

• EZ2Find - searches through eight search engines, and query results are accompanied by unique and useful features such as relevance indicator, preview and translation. EZ2Find also features clustering analysis, which organizes query results into a subject hierarchical structure, so that the various subject directories that the query results fall under, are easily identified through the visual display.

Although Meta-search engines are able to search through a wider coverage of web pages, but their performances are largely dependent on their associations and agreements with various search engines as the source. Also, developers of such search systems do not require an understanding of the back-end information retrieval process. Therefore, although information seekers could utilize Meta-search engines to perform searching tasks, when referring to information retrieval systems throughout the thesis, Meta-search engines are usually excluded due to its lack of the retrieval aspect. Meta-search engines are only included when a less formal and more general terminology of ”search systems” is used.

2.3.3 Web directories

Web directories are search systems maintained in a semi-automated manner that organize approved web pages in a hierarchical structure. Web directory is the search system that requires the most interaction and decision making from users, as instead of typing the query term, users are given an opportunity to browse for the information that they are seeking by clicking through a hierarchy of categories and sub-categories. As a result, with few clicks, users are able to locate information that is useful and have a certain level of quality.

However, the higher quality web directory system does have drawbacks as well. The drawbacks include the following:

• A limited number of web pages are indexed

Due to the manual involvements required, web directory usually indexes a limited number of web pages that in some categories, no result can be displayed [50]. In fact, it was quite common for some categories to have no result set at all, which the web directories later
solved by incorporating result lists from other search systems for those categories with no manually approved result.

- Expensive and resource intensive to maintain

This is because the high level of quality result is achieved by a group of human information experts filtering through web pages. To perform such a task manually to a large scale, or to do it repetitively to reflect the dynamic web is simply too time consuming and resource intensive [47, 114].

Therefore, besides the limited number of indexed web pages, the web pages indexed by web directories are rarely removed or re-evaluated due to the high manual cost of maintenance. As a result, the usefulness of web directories for searching on the dynamic World Wide Web is gradually decreasing, even though the quality of their search results remain higher than search engines in general.

Yahoo! was the first web directory system, introduced in 1994, and remained the most popular web directory [50]. Other web directories include about, look smart and open directory, where Open directory attempts to overcome the size limitation of web directories by encouraging web users to participate in the web page evaluation process. This directly increases the amount of manual resources by a large magnitude. However, the increase in manual processing still cannot match the number of web pages indexable by search engines. Also, increased manual involvement by the general public exposes the system to vulnerabilities such as bias and spam.

From the preliminary investigation into the major types of search systems on the Web, it can be concluded that search engine’s ability to index a large quantity of web pages through an automated crawling process is most useful for discovering information and resources on the gigantic and dynamic Web. Whereas the manual filtering process of web directory offers higher quality assurance on the search results. This leads to the speculation that perhaps an automated information retrieval and filtering process that imitates the higher quality manual processing in web directories, could accommodate the large and dynamic nature of the World Wide Web, and yet still offer an acceptable level of quality in the search results.

For the development of an information retrieval system, the identification of an appropriate crawler is perhaps the most important task; as the crawling strategy determines the system’s coverage and its view of the Web. A crawler is responsible for traversing the Web and retrieve web documents, so that they can be downloaded and stored in an information vault. The next section will discuss the crawling models that can be utilized by information retrieval systems.
2.4 Crawling models

Crawler is also commonly known as bot, ant, worm, web robot or web spider [77]. It is one of the core components of information retrieval systems, as it works in the back-end to retrieve data from the World Wide Web, so that information retrieval systems have the opportunity to process and index the web pages. The indexing process subsequently enables efficient search for matching web pages for a submitted query, producing a result set which can be displayed to the user.

There are a number of crawlers publicly available, with a large variation in the crawling features. However, due to the vast amount of data on the web to be retrieved, one common aspect of crawling which is continuously being improved, is efficiency. The crawling approaches can be categorized into four main models: basic crawling, focus crawling, parallel crawling and distributed crawling. The following sub-sections will describe each model and provide examples of systems that employ the different models.

2.4.1 Basic crawling

The basic crawling model retrieves all data without discrimination. It is the simplest of all crawlers to implement, and forms the basis of other models. There are two general approaches for basic crawling: breadth-first and depth-first. The two approaches differ in the order that web pages are retrieved, but theoretically, both would eventually obtain the same set of web pages.

- Breadth-first approach - Crawlers retrieve all neighbouring pages of the initial set of web pages first, before proceeding to the neighbour’s neighbouring web pages. Illustrating this crawling approach on a structure such as the one in Figure 2.6 will result in web pages crawled in the following order: $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G$
- Depth-first approach - Crawlers retrieve web pages by following the newly discovered links that point to a neighbouring web page, and continue deeper into a graph until a leaf page in a structure is reached or if all neighbouring web pages have been crawled. Then the crawler crawls the next sub-graph. Illustrating this crawling approach on a structure such as the one in Figure 2.6 will result in web pages crawled in the following order: 

\[ A - B - E - F - C - D - G \]

Systems that adopt basic crawling usually store a large repository of web pages, and index the pages according to keywords or topics with some ranking information. When a user queries the system, the indices are searched through, and pages with matching keywords or topics are identified as the query results. Those results are usually sorted in the order of their rank, and only a few are displayed per page.

Information retrieval systems that utilize the basic crawling model began at the first generation of search engines, and is still being used as a basis in current search engines. At the early stage of search engine development, a basic crawler was sufficient. The basic crawling model is simple to implement and maintain. However, when the number of web pages indexed by search engines increased to compete for the largest index or to maintain a significant portion of the web, the crawling efficiency of the basic crawler is no longer sufficient for independent use. Another model called focus crawling, is therefore developed to selectively retrieve data from the World Wide Web, narrowing the crawling target from the entire web to only a selected group of web pages.

### 2.4.2 Focus crawling

Focus crawlers are vertical crawlers that could refer to ”focus crawlers”, ”focused crawlers” or ”topical crawlers”. Throughout the thesis, ”focus crawler” will be used to refer to vertical crawlers in general, and ”topical crawler” will refer to vertical crawlers that focus on retrieving web pages related to a topic. Focus crawler selectively retrieves a set of target web pages, therefore is able to avoid the retrieval of less significant web pages, minimize the wastage of resources, and maximize the rate at which useful web documents are collected.

A focus crawler incorporates a mechanism to guide the crawler towards target pages, and to determine whether an un-visited page should be retrieved. The decision to crawl a page is often based on the information available about the page before retrieval, to arrive at an estimation on the benefit of retrieving a page. Focus crawling can be executed in real-time to the user query due to its ability of retrieving a large percentage of target pages after crawling only a small number of web pages.
Although focus crawler could be used to retrieve target web documents efficiently, it a drawback known as “tunnelling.” The term “tunnelling” refers to a process of finding a path between two distinct clusters of web pages which both meet a given criterion. In other words, it is possible that there is no direct link from a group of documents which meet a search criterion to another group of web pages meeting the same search criterion. Consequently, some non-target pages are intentionally crawled in order to retrieve other target pages. Although tunnelling decreases the rate that target pages are retrieved, it is necessary when a portion of target pages are only known to the crawler and therefore only accessible, after crawling some non-target pages. In most cases, focus crawling is still reasonably efficient, even after performing tunnelling.

Focus crawling has been used mostly to identify groups of web pages that address a specific topic, or web pages that satisfy a ranking threshold. There is currently a gap in literature in regard to focus crawling that is guided by some quality indicators. There are three general approaches of utilizing the focus crawling method, and they are listed and explained below.

- **Link-based focus crawling**

  This approach utilizes the link structure of web pages to indicate the usefulness of retrieving a page. For example, if many web pages contain a hyperlink to page A, the child page A may be a useful page. If most of those parent pages address a similar topic, chances are, page A is also an useful page to retrieve for the topic. This approach is quite challenging sometimes, as a page that is pointed to by many pages may not necessarily lead to another popular page that is linked to by many, therefore the rate at which useful pages are retrieved may not be high compared to other focus crawling approaches.

  An example of this type of focus crawler is the topical crawler discussed in [130], where the focus crawler is used to retrieve pages related to a given topic. The particular focus crawler addresses the tunnelling issue by proposing to probe the neighbourhood around a known group of pages by crawling pages which are not more distant than \( n \) links from the known group of pages, where the \( n \) link refers to the links which are contained in the \( n \)-th neighbourhood of the node of interest. The crawler then selects the most promising direction based on an assessment of the neighbourhood. It is shown in [130] that the approach is effective if two disjoint groups of relevant pages are not more distant than a distance of 3 links.

- **Anchor text based focus crawling**

  This approach uses anchor text associated with a hyperlink to provide clue as to whether the child page pointed to by the link would be useful. Before the concept of utilizing anchor text was introduced, there was no way of knowing
which phrase out of the entire web page is the description about the hyperlink, as the description could be placed before or after the link, or at the beginning or end of the page. Using the entire page is not useful too, as a page usually contains multiple hyperlinks with each of them addressing a slightly different topic. Therefore, the use of anchor text is especially useful for topical crawling, as crawlers could analyze the anchor text to provide more accurate prediction about the topic area of the child page.

The focus crawler developed by [101] is a topical crawler that is guided by Support Vector Machine (SVM) with a first degree polynomial linear kernel, to crawl web pages related to a given topic. Link context information is supplied to SVM through positive and negative examples for learning, and it includes the anchor text of hyperlinks, the entire vocabulary of the web document and the HTML tag structure. A varied amount and combination of information is evaluated for their effectiveness in predicting the benefit of retrieving an unvisited web page. The authors found that using both link context score and a full page score resulted in high precision topical collection, while using a tree representation of the HTML tag structure allowed topical collections with high coverage.

- Algorithm based focus crawling

This approach is relatively new. To date, focus crawlers have been used mostly to retrieve web pages addressing a selected topic area. There appears to be a gap in the literature as there has been no attempt to incorporate a scoring algorithm, similar to one that could be used to rank query results, into a focus crawler. The reason for this may be that a prediction needs to be made in order to correctly determine whether a web page is worth retrieving, and predicting or estimating a score based on information available before a page is retrieved is a challenging task. It is quite possible that, there is little correlation between the score of a web page and the score of the child pages it links to.

The Multi-agent focus crawler in InfoSpider [89] is a focus crawling approach which attempts to increase crawling efficiency by deploying multiple crawling agents simultaneously. The process is initiated by a set of keywords and a set of starting seed pages. Each agent start with the seed pages and performs focus crawling by evaluating the link values, which is assessed using reinforced learning method, with contextual words as input. Users can provide relevance feedback to assist the learning process further. This focus crawling application is one of the few that has been incorporated into a search system.

Focus crawling is very different to basic crawling, and it has many challenges, such as requiring an accurate prediction of the usefulness of retrieving a page, and the possibility that a useful
page may not always lead to another useful page. As a result, extensions from a different perspective were developed, so that multiple instances of basic crawlers can be executed simultaneously to improve crawling efficiency instead, in the form of a parallel crawlers.

2.4.3 Parallel crawling

Parallel crawlers under the parallel crawling model are developed with the aim of improving the efficiency of crawling the ever-increasing web, by executing multiple crawlers in parallel. Parallel crawling refers to multiple crawling processes executing simultaneously on a machine, or a group of machines such as a cluster system. The crawlers executing in parallel could be basic crawlers, focus crawlers or a combination of both.

In a crawling task, the bottleneck is usually the bandwidth of the network between the current machine and the server which hosts the target data; this means that CPU cycles are usually free for other tasks. This model was introduced to utilize the free CPU cycles during crawling by enabling multiple crawling processes in a machine to share the CPU resources without affecting the performances. Parallel crawlers often crawl different domains in parallel (i.e. $n$ parallel crawler processes crawl $n$ different domains). This is done to reduce the impact of the potential bottleneck between crawler (the client) and the Web server.

Although this model appears to perform well while making better utilization of CPU resources and bandwidth; however, executing multiple crawling processes on a machine or a group of machines imply that the valuable network resource is being shared as well. As a consequence, machines which host parallel crawlers require a fast network connectivity. But since even the fastest networks available (i.e. gigabit networks) are limited in capacity, this implies that the scalability of parallel crawlers is also limited. To further improve the scalability of crawlers, a new model called distributed crawling was developed by dispersing the crawling processes across several machines that could be located in various geographical locations.

2.4.4 Distributed crawling

Distributed crawlers are extensions of the parallel crawling model. Distributed crawlers also aim to increase crawling efficiency by executing multiple crawling processes simultaneously, however, the difference of distributed from parallel crawler is that the multiple processes are dispersed geographically instead of being in the same machine or group of machines. The distributed crawling model has several benefits.

- Crawlers do not have to share the same network, therefore crawling throughput will not decrease significantly with the increase of crawling nodes.
Machines in various geographical locations have the potential of conducting local crawling (i.e. the crawling of nearby sites and domains), which decreases the expensive and time-consuming international traffic.

A failure would not affect the entire crawling task. For example, power failure or a natural disaster in an area may affect only a portion of crawling processes instead of all of the crawling processes.

Variations in distributed crawling applications arise from the difference in management style of the crawling components. For example, the network topology could be managed in a centralized or decentralized manner. The two crawling management styles both perform more efficiently than using a single independent crawler. Assuming that the slaves are located on different machines, and have similar network bandwidth, the throughout improvement is theoretically as many times as the number of slaves, but taking away the communication overhead.

For the centralized network topology such as one illustrated in Figure 2.7, a master-and-slave arrangement is adopted where the master component is the central node that manages the slave components, and the slave components are dedicated crawler nodes. The only mean of communication is between the central and crawling nodes, no communication between the crawling nodes are carried out. The benefit for this arrangement is that the central node handles all management responsibilities, therefore minimizing confusion over the seed distribution and the crawling progress of crawling nodes. Also, since all crawling nodes only need to report to one destination - the central node, therefore, the crawling nodes can focus on the crawling tasks. Conversely, the centralized management style does have a weakness, which is the vulnerability of the central node. If the central node fails, the entire crawling process could be jeopardized.

For the decentralized network topology as illustrated in Figure 2.8, a peer-to-peer arrangement is adopted. In a peer-to-peer arrangement, there is no central node, but instead, information is passed through other crawling nodes. Each crawling node would have mechanisms to handle
the management information, as well as conduct crawling. Communication can be achieved by either passing a piece of information through the nodes in one direction, resulting in a ring formation, or by connecting each node with all other nodes. The benefit of this arrangement is that if any of the nodes fail, only the task carried out by the particular node is affected. Also, since each crawling node has the capability to process information, the crawled data does not need to be sent over the network to other nodes, only the much smaller sized status information is sent instead. The disadvantage of this management style is that the management tasks required by each node increases dramatically with the addition of new crawling nodes.

The current research project adopts a combination of models, namely focus crawling and distributed crawling models, and takes advantage of the key characteristics of these models. Although there is no restriction preventing the combined used of a distributed and focus crawler, however, there is currently no crawler, to the best of my knowledge, that merges distributed crawling with focus crawling. Therefore the current research aims at closing the gap in this research area by investigating into the possibility of merging these crawling models. The result is a practical crawler that efficiently retrieves web documents, and the implementation details are included in Chapter 7.

**Existing applications**

Distributed crawlers are developed with the aim of improving the efficiency of crawling the ever-increasing World Wide Web. They are basically multiple crawlers executing in parallel, but with the possibility of being dispersed geographically, so there is no need to share computing or network resources. However, variations in different applications arise from the difference
in management style of the crawling components executing simultaneously. Various features that were implemented to further improve crawling efficiency and assist with specific tasks also add to the separation. Here, four information retrieval systems that adopt distributed crawler are compared to show the various focus in the implementation of a distributed crawler and the differentiating features. A short description of each considered crawler is provided, highlighting its pros and cons.

- **Mercator**
  This crawler focuses on efficient crawling by implementing a sophisticated duplication avoidance mechanism and using DNS caching [64]. The arrangement of distributed crawling components is not as flexible as desired as the number of simultaneous crawling components has to be decided before crawling, and cannot be increased or decrease during the course of a crawl.

- **C-Procs**
  Developed by [29], this crawler focuses on providing flexible partitioning rules for the distributed slaves. Such rules can be URL, site or hierarchically based. However, static assignment of URLs to the distributed components before crawling begins, prevents its crawling flexibility and learning capabilities.

- **WebFountain**
  Developed by IBM [42], this was designed as an incremental crawler that allows crawl scheduling. This distributed crawler has a flexible distribution management, but its implementation is not platform independent.

- **Ubi Crawler**
  This is a decentralized peer-to-peer crawler that is system independent. Decentralizing the crawling process ensures low network overheads as crawled data can be processed locally instead of sending to a central location, but increases the system complexity, especially when errors occur [17].

In most cases, crawlers are developed with a number of focuses, so features that are considered less important may be absent in the crawling application. However, there are a few features that are important for practical reasons, especially for the tasks required for this research project. These features are compiled by reviewing the literature for the characteristics that should be displayed by crawlers that would be appropriate for the current World Wide Web
Table 2.1: Crawler performance as measured by practical features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Mercator</th>
<th>C-Procs</th>
<th>Web Fountain</th>
<th>Ubi Crawler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple and generic</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>System independence</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>O</td>
</tr>
<tr>
<td>Low communication overhead</td>
<td>-</td>
<td>O</td>
<td>-</td>
<td>O</td>
</tr>
<tr>
<td>DNS caching</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>O</td>
</tr>
<tr>
<td>Adaptive crawling</td>
<td>-</td>
<td>-</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>Local crawling</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Fault tolerant</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>O</td>
</tr>
<tr>
<td>Failure recovery</td>
<td>?</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

[17, 20, 29, 30, 42, 64, 116]. Table 2.1 shows how the mentioned crawlers measure against these practical features. The Table shows that a number of these essential features are not addressed by a majority of crawlers. Discussions about the implications of lack of attention to some features are explained in the paragraphs that follow.

All four distributed crawlers lack the feature of local crawling. In other words, all four considered crawlers do no have the ability to detect from the information provided by the web pages the geographical locality of the domains. This can generate significant amount of network traffic if a number of crawlers are used to crawl domains which are not geographically localized, eg. in the same region, in the same country. On the other hand, with local crawling, ie., crawling domains which are geographically close, it will reduce the amount of traffic generated on the Web. This aspect is important especially for servers which are located in networks that have relatively slow interconnect links with other Web domains, eg. other regions, or countries. Some crawlers allow the partitioning rule to be at the domain level, however, partitioning by analyzing the Top Level Domain (TLD) does not ensure accurate allocation of URLs for local crawling, as the TLD part of a domain name does not identify the geographical location of a server. Furthermore, a server may host various domains with different TLDs.

Also, all four crawlers lack the ability to recover from failure. A crawl of web can take weeks to complete or may be executed continuously. It is surprising that the crawlers do not seem to have an efficient mechanism to allow crawling to restart from where it left off, in the event of an unexpected failure. The mechanism adopted by Mercator for recovering is partially complete. It works by regularly writing snapshots of its status to disk, at the frequency of once a day. As a result, the crawling is able to recover, but some overlap or missing data might be encountered.
Three out of the four crawlers examined lack the following feature: the use of adaptive
crawling methods. The network bandwidth has been identified as a major bottleneck for crawling
tasks, but the unpredictable nature of the network in terms of possible traffic congestion also
affects the performance of crawlers. To allow appropriate use of the available network resources
without overloading the network and affect other users of the network, crawlers need to be able
to dynamically increase or decrease the number of parallel crawling components according to the
traffic conditions of the network, throughout the entire crawling process.

Besides these features that most distributed crawlers fall short of addressing, there are other
essential features not addressed by a minority of crawlers. In general, it can be stated that cur-
rently available distributed crawlers, as represented by the four considered in this chapter, do not
offer a practical set of features necessary for general purpose web page retrieval tasks. Therefore,
the crawler for the project was developed with these practical issues in mind.

2.5 Scoring and ranking models

Besides crawling, the scoring and ranking process is the other essential component of a search
system. Scoring refers to the process of associating a score to a web document. This score does
not change unless the scoring scheme or the content of the web document has been modified, and
the score is also not dependent on the query term used for searching. Whereas ranking refers to
the strategy used to order the search results for a submitted query, the ranking order is the order
that results are displayed to the user.

The scoring and ranking tasks are important in information retrieval systems, as is shown
in the literature [9, 12, 111, 121]; users expect desired information quickly, therefore normally
only click through and investigate results listed in the top 10 of the first result page, instead of
fumbling through thousands of results that match the query term. Therefore it is a goal for search
systems to provide web pages that meet the information needs of the user as the top results for
corresponding search queries. There have been many attempts to develop an algorithm that is
able to consistently rank user’s desired information highly in the result set. However, this has
proven difficult due to the various information seeking purposes of web users, and the common
use of search query terms that are insufficiently long and vague [12, 22, 111, 69, 77, 121].

The scoring and ranking algorithms implemented in existing information retrieval systems
can be divided into three major categories, and they are link based algorithm, usage based al-
gorithm and profile based algorithm. The categories are explained below, with the examples of
algorithms in each category immediately following the description.
2.5.1 Link-based algorithms

Link-based algorithms are ranking algorithms that make use of the link structures among web pages. Features that provide as a basis for link-based algorithms include the following.

- **Inlink** - The number of web pages that contain hyperlink pointing to the particular page.
- **Outlink** - The number of hyperlinks on the particular page pointing to other pages or to itself.

The inlinks and outlinks can be categorized finer into external inlink, external outlink, internal inlink and internal outlink, where external inlink and outlink could indicate that the hyperlink being of a different domain or site to the web page containing the hyperlink. Extensions could also include the anchor text surrounding a hyperlink. In which case, the content of the anchor text would determine the weight of score being assigned to the child page pointed to by the hyperlink.

Two of the most well known link-based algorithms that have been implemented into existing search systems are provided as examples[18, 41].

- **PageRank in Google** [http://www.google.com](http://www.google.com)

  Google adopts the PageRank algorithm as its main algorithm of assessing the importance of a web document. The pagerank algorithm is based on the referencing system of academic publication. In academic publication, the frequency that a paper is referenced by others is assumed proportional to its quality, as being included as a reference signifies the paper’s credibility and usefulness for other paper. PageRank is built on the same ideology, but the number of links pointing to a page (inlinks) is normalized by the number of links in the page (outlinks) to prevent web pages from excessive linking.

  PageRank $X \in \mathbb{R}^n$ is computed as follows, where $W$ is an $n \times n$ matrix with elements $w_{ij} = \frac{1}{h_j}$ if a hyperlink from node $i$ to node $j$ exists, and $h_j$ is the total number of outlinks of node $j$; otherwise $w_{ij} = 0$ [109]. Then the PageRank vector $x$ is a $n$-dimensional vector, and $n$ is the number of known Web documents. PageRank is computed by the following iterative procedure:

  $$ x(t + 1) = dWx(t) + (1 - d)1, $$

  where $d \in [0; 1]$ is said to be a damping factor, and $1$ is a $n$-dimensional vector with all elements set to 1. Although the rational for PageRank seems reasonable, however, the web environment is essentially very different to academic publishing environments. First of all, links in a web page is not deemed as significant as a reference in an academic paper.
Secondly, the content on web pages changes frequently and unpredictably, which means, the content of a web page may change after being linked, therefore the quality of the web page may also be changed. Lastly, this type of link analysis would mean that newly created web pages (which initially have no inlinks at all) will mostly remain lowly ranked as its existence is rarely known, so the number of links to it will remain low, resulting in a low Pagerank, which places the page towards the end of query result lists. While at the same time, popular web pages become increasingly popular as its position at the top of the query result list ensures its exposure to users. Therefore, PageRank is considered an evaluation of popularity rather than quality. Furthermore, the Pagerank algorithm can be vulnerable to manipulation by spamming pages.

Ever since Google advanced to the most popular and most used search engine, it has become a target for abuse. There are organizations with the sole purpose of manipulating Google’s ranking algorithm. For example, Pagerank is based on the assumption that a page linked by many sources is likely to be a good page, therefore, a link farm could be created as a way of manipulate the number of links pointing to a page (inlinks), so that a spamming web page could be placed higher in a list of query result. Link farm refers to a group of web pages containing many links to spamming pages or each other, for the purpose of harvesting a ranking that is higher than deserved for the spamming pages.

Another method of manipulation is based on Google’s use of text around a hyperlink (anchor text), or words in the page title to indicate the keywords for a page. As a result, spamming pages could respectively modify the anchor text within the link farm or insert a long title with many common search terminologies for the spamming page, so that the page will be displayed for a certain query, or be in as many result sets as possible.

As discussed, Google’s Pagerank seems to measure popularity rather than the quality of a web page, and seems to discriminate new web pages. Also, the algorithm is vulnerable to manipulation by spamming pages. Therefore, Pagerank is not appropriate for the task of identifying high quality web pages. This will be demonstrated in chapter 8 through a direct comparison between document quality and PageRank.

- HITS in Clever

The Hypertext-Induced Topic Search (HITS) algorithm is based on the work by Kleinberg [74], which assigns a hub score and an authority score to each web page according to its inlinks and outlinks. The intuition is that good hubs are pointed to by pages with good authority, and a page with good authority is pointed to by good hubs. The procedure proposed
by the article involves two steps: sampling and weight propagation. The sampling process collects web pages that are likely to be rich in relevant authorities using the following steps:

1. Gather a group of "root page", which are web pages relevant to a given query, and that could be obtained by querying a simple text-based search system.

2. Form a larger "base set", by including web pages that the pages in the root set point to (their child pages), and those that point to web pages within the root set (their parent pages). This base set usually has a size threshold.

From the collected base set, the authority weight \((x_p)\) and hub weight \((y_p)\) of page \(p\) are estimated iteratively through the following steps:

1. Construct an adjacency matrix \(A\) of \(n \times n\) where \(n\) is the size of the base set. The \((i, j)\)th entry is equal to 1 if page \(i\) links to page \(j\), and is 0 otherwise.

2. Then write the set of all \(x\)-values as a vector \(x = (x_1, x_2, ..., x_n)\), and similarly define \(y = (y_1, y_2, ..., y_n)\).

3. The update rules for \(x\) and \(y\) are as follows

\[
x \leftarrow A^T y \quad \text{and} \quad A x = (A^T A)x
\]

\[
y \leftarrow A x \quad \text{and} \quad A A^T y = (A A^T)y
\]

4. The vector \(x\) is the same as the result of applying the power iteration technique to \(A^T A\) after multiple iterations.

5. Finally, the output for a given search query are pages with the largest hub weights and the pages with the largest authority weights.

HITS is believed to perform better than PageRank, as it overcomes the difficulty of identifying similar natured and similarly important web sites or web pages, that due to competitive reasons, do not link to each other.

The study by [4] revealed that the correlations between the scores of various link-based algorithms are very high. All achieving relatively high performances when evaluated against results obtained from human experts. This may be attributed to the empirical finding [4] that incorporating a link-based ranking is usually able to improve searching and ranking. However, the ranking order of top 5 and top 10 web pages when compared against manually developed ranking list did not perform as well. Additionally, analysis from a different perspective by [82], found that
rank analysis based on link connectivity correspond to thematically related collections, in which most pages only address one dominant topic of interest. Therefore, web pages with a wider, and more general coverage of topics are discriminated and not considered as important as web pages of similar quality, but focuses on a specific topic. These characteristics of link-based ranking algorithm indicate that such an approach is not appropriate for a fair ranking of web pages, that reflects their quality.

2.5.2 Usage-based algorithms

Usage-based ranking algorithms usually require some logging of user behaviour to re-arrange the order in which search results are displayed, without identifying individual users. Therefore there is usually no requirement for user registration or profile establishment. The algorithm achieves personalization by firstly assigning an initial score to web pages, then identifying the ranking position of web pages in the result list and finally comparing that to the web pages that users choose to click through, to determine whether a web page should be ranked higher or lower. There are two general approaches for usage-based algorithms.

- Global update - The log is recorded on the system side, and the effect of one user’s choice of results to click through, would be reflected in the consequent search result of all other users.

- Local update - The log may be recorded on the user’s side, in the form of cookies, as an attempt to personalize results to match user’s interests. It is also possible to use information about the user’s machine, such as locality and operating system, instead of keeping a log. As a result, different users would have a set of query results that are ordered uniquely for them or for the group that they belong to. An example would be to display search results for Australian users, where local information would appear higher in the result list without the need for users to specify a local search.

The adoption of usage-based algorithms is not limited to information retrieval systems; it can be seen in many other Web applications, such as for servers that dynamically generate a page or for personalized Web advertising. Its common usage through cookies is often undetected by inexperienced users as no extra effort is required from the user. However, some users are concerned over its privacy implications as it often collects information about user activities without the users being aware.

- Boosting algorithm in Direct Hit http://www.directhit.com
Direct Hit runs an analysis on the results that match a user query based on the number of visits that the web page received from users. This can be interpreted as popularity as evaluated by information seekers, without the use of profiles to identify individuals. Direct Hit incorporates a boosting mechanism to personalize query results and to counter-act the attempts of rank manipulation [50].

The boosting mechanism works by considers the rank of a web page in a search result list and the number of users that clicked it. For example, a query result listed as top 5, but was not clicked by a user would have a lowered score; whereas if a user clicked on a lowly-ranked web page displayed as perhaps the 256th result, its rank would increase dramatically. The rank change would be effective the next time the web page is listed as a query result, and if other users do not click on the web page that has its rank boosted, then it will drop back down the result list for subsequent information seekers, as this would indicate that the web page has not proven popular.

An algorithm similar to the usage-based scoring and ranking algorithm is the profile-based algorithm. For users concerned over the logging of their activities on the web for use by usage-based algorithms, they could use a profile system instead, where users are able to log off their profile when they do not wish their activities to be logged.

2.5.3 Profile-based algorithms

Systems that adopt the profile-based algorithm maintain user profiles by imposing registration for the usage of its features, in order to identify and group users. Then personalization of search result is achieved through recommending the results that users with similar interest or behaviour consider useful [122]. Systems utilizing this type of algorithm require more effort from the users, as they are expected to establish a profile and regularly provide input on what they consider to be "good", in order to obtain useful recommended results. This type of system could only be effective with a large number of active users.

Profile-based algorithms are used more frequently for special purpose information retrieval systems such as for e-commerce, than in the general web-based searching. Profile-based algorithms is believed to out-perform usage based algorithms because of their profile establishment requirement that ensures all actions correspond to a profile. This means that users can be held accountable for their actions, and profiles that misuse the system can be blocked from further activity. Profiles also reveal the knowledge area and interests of an user, therefore indicating the amount of expertise that the user has in a particular topic.
• algorithm in Open Directory Project http://www.dmoz.org

The Open Directory Project (ODP) require all users who wish to contribute to the web directory to set up editor profiles. Editor profiles are obtained through an application process, where candidates should demonstrate their edition abilities or provide evidence of their ability to obey the ODP editing guidelines.

ODP’s editing model is a hierarchical one. Upon becoming editors, individuals generally have editing permissions in only a small category. Once they have demonstrated basic editing skills in compliance with the ODP editing guidelines, they then are able to apply for additional editing privileges, in either a broader category, or in a category elsewhere in the directory. Over time, senior editors may be granted additional privileges which reflect their editing experience and leadership.

All ODP editors are expected to abide by ODP’s editing guidelines. These guidelines describe editing basics: what types of sites may be listed and which may not; how site listings should be titled and described in a loosely consistent manner; conventions for the naming and building of categories; conflict of interest limitations on the editing of sites which the editor may own or otherwise be affiliated with; and a code of conduct within the community. Editors who are found to have violated these guidelines may be contacted by staff or senior editors, have their editing permissions cut back, or lose their editing privileges entirely. ODP Guidelines are periodically revised after discussion in editor forums.

ODP allows site submission without any charge, which differentiates itself from other web directories. The free site submission has two effects.

- One result has been a gradual divergence between the ODP and other directories in the balance of content. The pay-for-inclusion model favours those able and willing to pay, so commercial sites tend to dominate in directories using it. (See for example the initial impact on Looksmart.) Whereas a directory manned by volunteers will reflect the aims and interests of those volunteers. The ODP lists a high proportion of informational and non-profit sites.

- Another consequence of the free submission policy is that the ODP has enormous numbers of submissions. The ODP now has approximately two million unreviewed submissions, in large part due to spam and incorrectly submitted sites. So the average processing time for a site submission has grown longer with each passing year. However the time taken cannot be predicted, since the variation is so great: a submission
might be processed within hours or take several years.

- algorithm in I-Spy http://ispy.ucd.ie

In I-Spy, instead of establishing profiles of individual users, profiles are established for a community of users instead. This is done to avoid privacy concerns where individuals can be identified and their searching activities recorded [35]. The past searching behaviours of other users in the same community or in other similar communities, is assumed to correlate due to the similarity in interest, and therefore is used to reveal the common queries and search results that the user is likely to find interesting or useful. Those useful queries or search results are then recommended to the user, similar to the usual recommendation system [27].

For the I-Spy system, the server records queries submitted and the pages selected in a hit matrix, then the Jaccard coefficient of term overlap is used to calculate query similarity. The ranking is in the form of a relevance score calculated using the following formula, where for a hit matrix $H$, the element corresponding to the $i$th row and $j$th column represents the number of users that selected page $p_j$ from a result list for query $q_i$ [35].

$$Relevance(p_j, q_i) = \frac{H_{ij}}{\sum_{v \neq j} H_{ij}}$$ (2.4)

The advantage for this approach is that logging from the server side ensures that users can experience personalized searching regardless of their location. Also, the the concern of lack of privacy due to the logging of individual user’s searching behaviour, is also decreased with the community based profile. However, since this is a relatively new approach to the traditional profile-based ranking, there is little empirical study that confirms the performance and reveals the effectiveness of this approach, as judged by users.

The scoring and ranking algorithms in existing information retrieval systems generally fall under one of the above mentioned categories. It can be seen from the category descriptions that ranking in most of the current search systems is usually carried out by sorting pre-calculated scores. This can be carried out using the following procedure:

1. Identify web documents that match a submitted query by searching through an index
2. Obtain the scores for those matching document
3. Sort the list of matching documents according to their corresponding score
It has been observed by [68, 142], that the use of a ranking algorithm, no matter the basis or approach, usually improves information retrieval results. This confirms the importance of having scoring and ranking algorithms in a search system. However, there is no algorithm adopted by search systems in practical use that addresses the issue of quality by identifying web pages of high quality from the others.

After the examination of existing information retrieval systems, it is clear that the back-end developments such as the crawler and a quality evaluation algorithm have not reached a standard that is able to provide optimum support for information retrieval systems. Such an observation is confirmed by [9, 60] through empirical analysis that the ranking standard of public web search engines is by no means state-of-the-art. Furthermore, as pointed out by [79], although link-based ranking algorithms are the foundation of many popular search systems, it is no longer sufficient to only use link-based analysis for the current World Wide Web. That is the reason that many current search systems such as Google and CLEVER incorporated a large number of unpublished heuristics (some may even require manual processing) in addition to their link-based ranking algorithms.

As a consequence, a novel crawler and quality evaluating ranking algorithms are required to advance in the information retrieval field. That is also the reason a novel crawling software with a quality evaluation algorithm was developed during the research project to provide quality information retrieval. The details about them are in the following chapters, Chapter 7 and Chapter 6.

2.6 Index update approaches

In addition to the vast amount of data on the web, the data also changes frequently, and the addition, modification and removal of data are often carried out unpredictably. Therefore, disregarding the crawling model employed, crawling is usually conducted periodically so that the changes on the web can be reflected in the information retrieval system’s index, thus keeping the query results relatively up-to-date with the current web. The following subsections outline the re-crawl and refresh strategies that could be employed by search systems to ensure that the search index remains relatively up-to-date.

2.6.1 Re-crawl strategies

In the past, a set of web pages would be crawled and then re-crawled periodically. In the re-crawl process, all web pages would be retrieved regardless of whether a modification to a page has
been made; however, with the increasing size of the web, more efficient and effective re-crawl strategies are now adopted to update the index. It is common practise now, to check the header first and only retrieve the contents of web pages that have been modified since the previous retrieval. The following outlines some major re-crawl strategies that can be adopted by crawlers depending on the priority of web pages that should remain up-to-date.

- **Total re-crawl**
  This strategy considers all web pages to be of equal importance in remaining up-to-date. The crawling of the entire set of web pages is only repeated after one full round of crawling has completed.

- **important-first re-crawl**
  This strategy is useful for search engines that sort query results according to some rank. Since users are more likely to access highly ranked web pages, they receive higher priority to remaining up-to-date. Therefore, this strategy focuses on updating highly ranked web pages by setting the re-crawl frequency of web pages proportional to their rank.

- **modification frequency based re-crawl**
  This strategy re-crawls web pages according to an estimated modification frequency. Web pages change at different rates, for example, news pages would become out-of-date much sooner than informational pages about a country provided by government bodies; therefore, search systems that allow news search are likely to adopt this re-crawl strategy. For this strategy, an investigation into how the web changes is required, in order to make a prediction on the rate of change for web pages.

### 2.6.2 Refresh approaches

Web contains a dynamic collection of data where insertion, removal and modification is continuously and unexpectedly being carried out. Therefore, after information is retrieved from the web and indexed with some score to allow system users to search through, considerations need to be given to ensure that new data crawled consequently through the re-crawl procedure can be inserted into the index, while there may be users accessing the search system at the same time. The approaches by which this can be done are described below [83].

- **Batch mode refresh**
  This approach waits until all new data are collected into a temporary location, then the old contents are replaced by the new contents in one swift move. Because the actual
replacement of old contents is an efficient file-based move procedure, users are less likely
to experience problem while using a system that adopts this refresh approach. However,
the downside to this approach is that since the system needs to wait until all new data are
collected before refreshing the content in the index, some web pages retrieved earlier in
the process and are required to wait, may already be out-of-date. Therefore, the freshness
of the content varies, depending on the time it takes to fully retrieve all new data and the
order that the new data are retrieved.

- real-time refresh

This approach replaces the old content as soon as its newer version is retrieved. This
ensures that at the time of refreshing, the new version is always up-to-date, allowing an
easier tracking of the update frequency of a web page. Also, this approach allows more
flexibility in the sense that each web page can be retrieved and refreshed at a different
frequency, which is useful for the “important first” and “modification frequency based”
re-crawl strategies. However, since the refresh process is executed as soon as a web page
is retrieved, the number and frequency of refreshes will be considerably large, therefore, in
implementing this refresh approach, care must be taken to ensure that users are not affected
by the constantly updating index.

After an overview of search and retrieval on the Web at the beginning of this chapter, back-
ground information on each of the basic back-end components of a search system have been
covered. The background information include models for crawling application, the various scor-
ing and ranking algorithms, and the maintenance strategy of a search index. Since this research
project aims at more than just information retrieval, but quality information retrieval from the
web, it is important to also review the literature to understand quality assessment approaches in
the past. Details of quality models are provided with some comparison in the next section.

2.7 Quality assessment models in the literature

Quality in a general context, is interpreted as the degree of excellence. Another interpretation is
from the understanding of data quality where quality refers to the “totality of features and char-
acteristics of data that bears on their ability to satisfy a given purpose” [135]. The interpretation
of quality as applied to this research project is from a more general context with no reference to
its intended use; the interpretation is in the sense that quality is only defined by the characteris-
tics of an entity, which is more objective. However, even from an objective viewpoint, quality
cannot be easily defined, as it is often a matter of human judgment [4]. For the purpose of this
research project, quality is interpreted as the value which the information provides to the user of that information, determined by its degree of excellence.

It is increasingly important to evaluate the quality of documents on the Web, as the lack of standard for web publishing and the number of documents on the web has made it challenging to obtain high-quality information. It was confirmed in the Digital Future Project, where users rated the credibility of information on the web as fairly high in the first three years, but a slight decrease was observed in the third year (2002). The credibility rating dropped even lower in the fourth year (2003), indicating that users perceive that the average quality of information on the web is decreasing [108]. In an informal interview with an experienced librarian as part of a preliminary investigation of the need for a quality-focused information retrieval system, the experienced library also observed the phenomenon of increasing difficulty to locate high quality information on the World Wide Web. The log from the informal interview is included as an appendix to this thesis (Appendix A). All these imply the need for quality filtering and assurance on the Web.

The investigation into information quality on the World Wide Web is actually not new, however, the increasing size and the dynamic nature of the web make such a task a challenging one [68, 69, 103], as quality measurement is not a trivial task. Quality relies quite heavily on the cognitive interpretation by manual effort, and it usually requires metrices of criteria as is shown in literature [26, 43, 93, 142], which may include criteria such as well-written, comprehensive, factually accurate and non-biased. These criteria can be challenging to measure both manually and mechanically. The task is made even more complex due to the following issues:

- The variation in user’s information seeking behaviours

The target information for searching is dependent on the purpose, and the intentions for information seeking can be categorized into the following

- Background search - Usually conducted when users do not know the topic area well. Users have a vague idea of what would constitute as the target information.

- known-item search - Usually conducted when users know the topic area or the method required to find a piece of information. Could also be used to verify a fact or to seek a site entry page.

- Decision task - Usually conducted when users need to find a wide range of information in order to compare or make an informed decision.

- Many items task - Usually conducted when users need to find multiple pieces of information on a topic, similar to seeking multiple answers for a question.
However, it is observed that users use short and undescrptive queries when searching, and often search using same terms for different searching purposes [69, 37]. The use of inappropriate query results in the search system unable to identify the search purposes, and therefore unable to present results that match the user’s information needs.

- The variation in the standard of information contained in web pages

There are evidence that some web pages contain wrong, incomplete or un-credited information. Also, there seem to be a trade-off between the breadth (coverage) and depth in the content. The analysis by [13] revealed that there are only few web pages that contain many facts, many web pages containing limited facts, and that no single web page or site provides all the facts regarding a specific topic. As a result, perhaps current search systems are more appropriate for known-item search [13], but not other information seeking purposes.

- The increase of web-based scam

The constantly increasing population with access to the World Wide Web and the number of online-based business imply that it is worthwhile to increase exposure using the Web as a medium. That could be the reason that popular services and web pages are often targeted and exploited for unsolicited advertising. These advertising could be in various forms, ranging from manipulating search engine’s ranking algorithms to appear highly ranked or to appear in numerous search results, harvesting email addresses for sending junk emails, posting unwanted information in message boards or forums, to hacking into web sites to post or obtain information without authorization.

The current ranking mechanisms incorporated into existing search systems attempt to address the issue of too many search results with varying level of quality, but instead of incorporating a quality evaluation algorithm, they mostly focus on hyperlink-based popularity instead. Although there appear to be an absence of quality evaluation scoring or ranking algorithm incorporated in practical search systems, but the literature offers extensive discussion about quality models and proposals of quality criteria, which can be examined to identify potential application to documents on the evolving web.

2.7.1 General quality models

Although the current quality assessment mechanisms in practical applications do not sufficiently evaluate the quality of web pages, but there are general quality assessment models in literature that could be investigated. The models included in this subsection are general models that form
the basis of most of the new models developed in recent times. The investigation below aims to study the possibility of extending and quantifying them for the task of assessing the quality of web documents.

• Data quality model by [133]

This is a conceptual model for data quality, consisting of 4 categories, namely intrinsic data quality, accessibility data quality, contextual data quality and representational data quality. Each of the categories contain dimensions to evaluate data quality, arriving at 16 dimensions in total. The dimensions grouped by their categories include:

– Accuracy, objectivity, believability and reputation
– Accessibility and security
– Relevancy, timeliness, completeness, amount of information and value-added
– Interpretability, ease of understanding, concise representation and consistent representation

This model is the benchmark and the foundation of most existing quality models.

• Software quality model by [139]

The approach is based on the ISO model and is for the evaluation of software quality. The model comprises of 6 quality characteristics: functionality, reliability, efficiency, usability, maintainability and portability. Each characteristic contain sub-characteristics, resulting in a grand total of 32 sub-characteristics. The sub-characteristics grouped by the 6 quality characteristics include the following:

– Suitability, accuracy, interoperability, compliance, security and traceability
– Maturity, recoverability, availability, degradability and fault tolerance
– Time behaviour and resource behaviour
– Understandability, learnability, operability, luxury, clarity, helpfulness, explicitness, customisability and user-friendliness
– Analysability, changeability, stability, testability, manageability and reusability
– Adaptability, conformance, replaceability and installability

Although a few of the sub-characteristics cannot apply to web documents, a small portion of them could still be used for the evaluation of document quality. Using software quality model to define and even increase the quality of a web page is also supported by [39].
These models are not developed with the purpose of evaluating the quality of documents on the World Wide Web, therefore cannot be adopted to assess the quality of web documents without modification. Although these two models are important basis for many of existing web-based quality evaluation models, but as [76, 113] pointed out, data quality attributes can vary depending on the context that it is to be used. The general models for assessing the quality of documents were not developed for the context of practical information retrieval on the World Wide Web. Therefore, the suitability of those quality assessment models for documents on the web is questionable as there are essentially two challenges for these general quality assessment models. The challenges are firstly, the difference in the information environment, and secondly, the difference in retrieval purpose.

The first challenge for the general models is that the information environment between traditional information retrieval and information retrieval on the web is considerably different. The web is different from traditional information retrieval environment in that it is open, constantly accessible, interlinked, extremely large, non-static, has no enforceable quality for information uploaded onto the web, has no retrieval standard, and is unsafe with component parts vulnerable to break down or attack [28, 75, 106]. Consequently, the general models fail at being able to be employed on a large scale, and on such a varied and dynamic set of documents.

The second challenge for the general models in literature is that the purpose of information retrieval is much more diverse on the web. Information retrieval on the web is no longer limited to information seeking, it can also include transactional and navigational purposes [20, 69]. The additional information retrieval intentions introduced by the popularity of the World Wide Web are not addressed in existing models in a fashion that can be implemented into a practical search system. Furthermore, the currently accepted information quality assessment involves evaluating quality from the user’s perspective. However, users of the information on the World Wide Web require information for a variety of reasons, yet may use identical query terms in a search system [12, 77, 111]. Therefore, making it difficult to differentiate the intention of information retrieval, and subsequently unable to determine whether the user’s information seeking objective was met.

2.7.2 Quality models for web documents

Since the general quality models examined in the previous subsection are not suitable for evaluating web documents, this subsection will examine the more recently developed models that take the characteristics of the web environment into consideration, and are developed specifically for web documents. These quality assessment models are not yet implemented into a practical search system, therefore the discussion below aims to examine the models for their suitability for the
task of measuring web document quality.

- Simple quality model by [142]

The quality model is based on relatively simple metrics, without large number of criteria. This model aims to enhance current crawling technology with logical algorithms that quantify characteristics. The quality metrics include currency, availability, information-to-noise ratio, authority, popularity and cohesiveness [126, 142], and are computed as follows:

- Currency - The last-modified timestamp of the document.
- Availability - The percentage of the links in the page being broken links.
- Information-to-Noise Ratio - The total length of the tokens after some pre-processing divided by the size of the document.
- Authority - Based on the score ranging from 2 to 4, given by Yahoo Internet Life (YIL) reviews, or 0 if it has not been reviewed.
- Popularity - The number of links pointing to a web page (in-links).
- Cohesiveness - Determined by how closely related the major topics in the web page are, by comparing to a reference ontology.

These quality metrics are selected from 16 quality criteria used by human evaluators for search services that provide a rating feature; examples of such services include Internet Scout, Lycos Top 5% and Internet Public Library (IPL) [142]. The selection of the 6 metrics from a total of 16 is based on their wide use across the considered search services and their potential for implementation to achieve automated analysis. The verification of the effectiveness of the model is performed by 4 participants who rated various search results as either relevant or not-relevant to the specified search term[142].

Although this model provides detailed description about the approaches for obtaining a score for the metrics, the metrics do not seem well-founded. Also, the verification process by the participants were verifying the relevance of the search results, rather than their quality, therefore, the effectiveness of using this model for quality assessment is yet to be verified. Additionally, the most important quality index for documents as pointed out in the literature [1, 76] - information accuracy - is not included as one of the characteristics. Therefore, the usefulness of this model is questionable. Nevertheless, some approaches suggested in this model such as currency and information-to-noise ratio are feasible to be applied in practical setting to a large scale.
• Three-tiered quality model by [93]

The approach is more complex, using a three-level assessment for the quality of an information source, according to the subjects, objects and processes involved in information retrieval. This model defines information quality criteria as follows, groups in their assessment classes.

- Believability, concise representation, interpretability, relevancy, reputation, understandability and value-added
- Completeness, customer support, documentation, objectivity, price, reliability, security, timeliness and verifiability
- Accuracy, amount of data, availability, consistent representation, latency and response time

Although this model would provide comprehensive information regarding the quality of a document from a user’s perspective, with its total of 22 quality criteria. However, this is not useful for incorporating into a search or retrieval system as implementations to allow automated evaluation was not suggested for these seemingly manually-intensive quality criteria. The writer went as far as pointing out that some of the criteria would have to be dependent on individual user’s experience, so an observation from a sample of users will not be able to be applied to all users, and that continuous assessment of information quality by the user would be required through the maintenance of individual score profile for each user. As discussed earlier, due to the large amount of data on the World Wide Web, tasks requiring manual resources or time-consuming tasks is not achievable, as it would be time-consuming to carry out manual processing on the web scale.

• Information quality measurement model by [43]

This approach used Information Quality Measurement (IQM) to identify potential information quality related problems associated with individual web pages, and then categorized them into information quality dimensions or criteria. The 16 quality dimensions are listed below, grouped by their quality type of either content quality or media quality.

- Accuracy, comprehensive, clear, applicable, concise, consistent, correct and current.
- Convenient, timely, traceable, interactive, accessible, secure, maintainable and efficiency.
The web applications that can be used as tools to measure the extent of the problems are also listed in this work. This model contains some useful criteria that is feasible for implementation into a search or retrieval system; however, a large number of criteria are not considered important by the user survey conducted as part of this research project, or are simply a “yes or no” type of question, which may not be very useful in practise. Hence, the main shortcoming of this approach is that the quality assessment criteria are not aiming at addressing real user needs but is making ad hoc assumptions on user needs.

- Algorithm combination model by [26]

This model uses an interesting approach, where no specific quality criteria are used, but instead, the well-known ranking algorithms in literature, such as TFIDF, Pagerank and HITS are included. These ranking algorithms are computed for the following aspects:

- The content of the page itself
- The content of the page’s neighbouring documents
- The page’s link information

This model then takes the result of the algorithms as input for the Support Vector Machine (SVM), which is used as a machine learning classifier. This model is highly feasible in incorporating into practical systems, which is dramatically different to other more theoretically based models mentioned so far. The shortcoming of this model though is consistent with the shortcomings of the ranking algorithms it is based on, which is the fact that quality is not directly addressed. For example, it will be shown in Chapter 8 that PageRank has no measurable correlation to document quality.

Although these models contain some possible criteria that can be considered in the evaluation of web documents, three questions need to be answered. Firstly, how can a quality assessment routine be tested? The difficulty of such a task is emphasized by the fact that none of the previously listed approaches show that the quality score is indeed agreeable to human perceived quality of a Web document. Hence, this thesis will develop a framework for testing quality automated assessment routines. Secondly, how important are each of the quality criteria? Are the criteria sufficiently important for the effort and time required to evaluate them? The importance of the criteria need to be defined, so that the criteria can be organized according to their importance, and only high impact criteria are incorporated into the crawler. Also, an indication about the degree of importance of each criterion, in comparison to other criteria, is required in order to correctly assign a weight to the scores for the criteria. There are literature [76, 103] that have
attempted to produce such a list, by identifying the common criteria that literature in the field found to be important for the evaluation of document quality on the web, in the order of the frequency at which they are mentioned.

The two summarizing works have however, evaluated a large number of common literature, therefore calculating the average of the two would not be a fair evaluation of the quality criteria according to their importance in the literature. As a consequence, the work by Knight[76] was used as a basis, and the literature from Parker[103] which were not included in the work by Knight[76], were analyzed independently, and added to form an updated statistical summary of the quality criteria. Table 2.2 is the summary, showing the number of times and the percentage that the criteria are considered important in the literature examined by the two summarizing works, where the third column from the left only contains the statistics from literature not already analyzed by Knight[76].

Table 2.2: Quality criteria ordered according to their overall importance as recognized in literature

<table>
<thead>
<tr>
<th>Quality evaluation dimensions</th>
<th>In literature examined by [76]</th>
<th>In literature examined by [103]</th>
<th>Combined total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>8</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>Timeliness</td>
<td>7</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Consistency</td>
<td>7</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Security</td>
<td>7</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Completeness</td>
<td>5</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Reliability</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Concise</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Understandability</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Accessibility</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Relevancy</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Availability</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Objectivity</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Useability</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Reputation</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Efficiency</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

As Table 2.2 indicates, accuracy referring to the extent that the information contained in a web page is correct and reliable, has found to be the most significant indicator of quality accord-
ing to the literature. It was recognised by 9 published work out of the 13 unique work examined through [76, 103]. Followed by the second significant criteria of timeliness, which refers to the up-to-dateness of a web page, and consistency, which refers to the similarity in the presentation format of information and the compatibility with previous data. Then security, meaning the extent that access to information is appropriately restricted. Completeness has identified to be the fourth most significant criteria for evaluating quality, and reliability, concise, understandability, accessibility and relevancy share the fifth position. Availability, objectivity, useability, reputation and efficiency fared equally well at the sixth position. These criteria are identified in 4 of the total 13 literature examined, which means that they are only recognised by approximately 31% of the literature. Other criteria such as reputation, amount of data and believability are recognized by even less number of literature, therefore are not included in this table.

It is interesting for relevancy to not be ranked in the top 3 criteria that are considered significant to web page quality. This is perhaps due to the perspective of quality by some literature, which believe that relevancy should be independent of quality, and be evaluated separately. This corresponds to the concept of quality as defined in this thesis. Although this table shows the order of significance for quality criteria, the weight is still unknown. For example, we obtained that accuracy is more important than timeliness, but what is still unknown, is by how much more.

Secondly, how can the criteria be incorporated into a practical application? A majority of the criteria are difficult to implement into a practical information retrieval system for evaluating web documents on a large scale. What practical approaches can be taken to address each quality criteria? Also, how do we know that the approach chosen is appropriate for the task?

These questions are challenging to address, therefore, a user survey is required to verify the degree of importance of the various criteria and the practical feasibility of quality criteria needs to be evaluated. In addition, due to the difficulty of the task, machine learning methods are also proposed to assist in answering the question of the appropriateness of the chosen approaches. The following section will provide some background information on machine learning to provide awareness of its purpose and capability.

2.8 Machine learning approaches to quality estimation

This research project proposes to adopt machine learning (ML) approaches to assist the development of a high quality information retrieval system. Machine learning approaches has been extensively researched, with successful applications ranging widely from robotic locomotives, gaming, speech recognition to stock market analysis. It has been indicated so far that a clear set
of “rule” or formula cannot be defined to evaluate the quality of web pages, therefore the capabilities of machine learning methods may be able to assist in defining the relationship between the variety of web documents and their perceived quality.

Machine learning is the autonomous acquisition and integration of knowledge. This capacity to "learn" from experience, analytical observation, and other means, results in a system that can continuously self-improve and thereby offer increased efficiency and effectiveness [6]. The terminology "learn" in this context refers to the process of using example data or past experience, to optimize a performance criterion or to build a general model, which is necessary when:

- Human expertise is not available
- Human is unable to explain their expertise due to the difficulty in defining rule sets
- Solution changes in time, or
- Solution needs to be adapted to particular cases

As shown, machine learning is useful for problems where relationships or sets of rules cannot be clearly defined. Machine learning approach is also able to apply the knowledge learnt from a set of data (training set) to other unseen data, which is often referred to as its generalization ability.

Artificial Neural Networks for example is one branch of machine learning where the task is to develop a model which simulates the ability of a (human) brain to learn. Such an approach has the obvious advantage of not requiring to hard-code information into a computer and thus, is much more flexible in its application. However, ML is generally known as an inexact science in that the answer of a ML method can only be approximately correct. In practice, this approximation is extensively exploited as it allows to obtain a system that remains stable in a noisy (learning and application) environment.

For example, ML methods are popularly applied in human-machine interfaces such as speech recognition systems. No two voice signals are every produced exactly identical due to the analogue nature of sound. Thus, a voice recognition system needs to be able to recognize every differing signals in a possibly noisy environment. A ML method is trained on a set of sample signals together with a desired output signal. An iterative training approach is commonly applied with the aim to adjust internal parameters such that for a given input signal the desired output signal is produced. Such learning methods are generally asymptotic which means that the desired output can never be reached accurately. However, most ML methods are proven to be able to
approximate any input-output relationship to any arbitrary precision which makes ML methods work very well in practice.

Machine learning has some challenges as well. The data that is used to train the network need to be appropriately constructed to represent the various categories of data. If the training data does not sufficiently represent the overall problem, it will result in poor performance, or may even experience the over-fitting problem. On the contrary, if the training data is too large, the machine may experience generalization problem, where excellent performance is obtained from data already seen, but unseen data results in poor performance.

Machine learning algorithms are organized into a taxonomy, based on the desired outcome of the algorithm. Common machine learning algorithm types include:

- **learning associations** - To find relationships in the data
- **supervised learning** - To learn a mapping from the input to the output where correct values are provided during training
- **unsupervised learning** - Only given the input data, to find regularities in the data
- **reinforcement learning** - To learn a policy that maps states to actions

According to [45], there are frequent use of machine learning methods in the information retrieval area, especially in the following tasks.

- **Text categorization** - where Support Vector Machine (SVM) or Bayesian classifiers are often used [102, 112]
- **Document clustering** - where unsupervised learning methods such as Self Organizing Maps (SOM) and SOM extensions are popularly used [72, 57, 114]
- **Information Extraction from documents** - where a combination of machine learning methods are used
- **Topic specific focus crawling** - where a variation of machine learning methods have been adopted to predict the benefit of crawling un-visited web pages [101, 130]

For this research project, machine learning methods are being used to assist the identification of high quality web pages based on the information that can be extracted from the web document. This is due to the fact that the human cognitive decision of determining the quality of a web document cannot be defined by a rule set that can be transformed into a machine implementation algorithm directly, which means the criteria used to judge the quality of a web document cannot
be clearly defined, even though [126, 132] showed that human experts tend to agree on which web documents would be considered to be of high quality.

As a result, the proposed machine learning method will be of type “supervised learning” since the high-quality web pages can be identified within the training set. With known output as a target, the machine learning method could assist in identifying the criteria that would be useful in evaluating unseen web pages, and to derive a weight to each criteria, so as to provide a clue of the page’s quality without additional manual processing.

### 2.9 Conclusion

The background information provided in this chapter are based on the research and developments in relevant areas before the research began. They provide a foundation to the research and also assist in identifying areas that require more study and exploration. It can be seen from this chapter that the design and development of an information retrieval system has been a focus in recent years. The investigation into quality also received a great deal of attention. However, there appear to be a gap in the area, as there has been very limited known work on the incorporation of quality evaluation into an information retrieval system. Instead, what can be found are either theoretical research on quality which cannot be readily incorporated into an existing search system for automated quality evaluation, or practical applications for information retrieval which do not directly address the quality of document content. This may be contributed to the difficulties of expressing the criteria used to evaluate the quality of a web page, and possible discrepancy of criteria used among web users.

Although challenging, a quality evaluation system needs to be in place in order to separate the high quality web pages from others, otherwise the small number of high quality web pages may be buried amidst the long list of search results, and may even be overlooked. This would result in web searching becoming increasingly frustrating for users. As a result, some search systems partially addressed the issue of a lack of quality control through the incorporation of a ranking scheme, but there is no proven correlation between the ranking schemes used by current systems and the actual quality of web pages. Therefore, a new approach is proposed, which utilizes machine learning methods to mimic the process of manually sorting and filtering through web pages in search of high quality web pages, but in a more automated manner.

As a first step in such a task, some data needs to be gathered for use in future experiments on a smaller scale. Once an appropriate set of data is collected, the identification process of high quality web pages can then begin. The next chapter will describe the data collection process of
this research.
Chapter 3

Data collection

3.1 Introduction

It is commonly known that the Web is enormous in size and dynamic in nature. Some literature attempt to estimate the size of the Web or observe the changes in the Web over a period of time, all of which revealed interesting properties. However, the size and dynamics of the Web can only be estimated; even the rate at which the web is growing or changing cannot be defined or projected precisely with certainty. This is a great challenge for many Web-based research projects, because this means that experiments conducted on the Web cannot be reproduced or verified.

The nature of this research is very much web-based, thus it would be beneficial to observe and understand the characteristics of the Web; as a better understanding of the Web would assist in designing the proposed quality information retrieval system to be well-suited for utilization on the World Wide Web. However, as experiments conducted on the Web cannot be reproduced or verified, there is a need for a static set of Web data that is reasonably large and a good representation of the Web for testing purposes. This is often done by retrieving a closed set of web data from the Web over a short period of time to ensure that the varying nature of the web does not introduce significant effect on the collected data. This collection of Web data is often referred to as a testbed, dataset, Web snapshot or Web repository.

There are numerous research areas which utilize a testbed to evaluate new proposed algorithms and approaches, as it allows reproducible experiments to be conducted within a controlled environment, and allows conclusions drawn from experiments on such a collection to be applicable on the World Wide Web. There are some existing testbeds available for this purpose, and they are described in the following section.
### Table 3.1: Number of papers which employ the WT10G and/or WT100G in experimental settings

<table>
<thead>
<tr>
<th>Year of publication</th>
<th># Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>1</td>
</tr>
<tr>
<td>2001</td>
<td>3</td>
</tr>
<tr>
<td>2002</td>
<td>5</td>
</tr>
<tr>
<td>2003</td>
<td>17</td>
</tr>
<tr>
<td>2004</td>
<td>17</td>
</tr>
<tr>
<td>Conference papers</td>
<td>40</td>
</tr>
<tr>
<td>Journal papers</td>
<td>3</td>
</tr>
</tbody>
</table>

3.2 Existing testbeds

Some existing benchmark testbeds were investigated for the possibility of being used for the experiments in this research. The number of openly available testbeds are limited, and the testbeds that the research team is aware of include WT10G and WT100G, the INEX 2005 Movies collection and the INEX 2006 journal paper collection.

#### 3.2.1 WT10G and WT100G benchmark testbeds

The WT10G (Web Track 10Gigabytes) test collection is a benchmark dataset which was distributed in 2000 by CSIRO in Australia, and presented itself as an ideal testbed for web-based Information Retrieval experiments. The popularity of the dataset is reflected in a preliminary investigation on published literature which identified 43 papers which employed the WT10G and/or WT100G data collection to evaluate their approaches, as shown in Table 3.1. The WT10G collection is a subset of the larger collection known as WT100G or VLC2, developed from a crawl of web content in 1997. The WT10G collection was constructed to retains the properties of the 1997 web content, which include the web link graph structure, server size distribution, the inclusion of inter-domain links and the inclusion of web pages on various subjects [59].

The WT10G and WT100G collections will be compared to the major characteristics identified in the literature, which include the basic characteristics, the page validity, the variation in web content, and the variation in the file type. The comparisons are means of measuring the usefulness of the WT10G collection for IR researches in the current setting.
**Basic characteristics of the testbeds**

Although the WT10G collection aimed to model the web, not all properties of the web data are reflected in the collection. Considerations were given during construction to eliminate undesirable documents that would affect information retrieval experiments negatively. For example, non-English documents, duplicated documents at the same URL, documents with identical checksums on the same server, and documents that are not HTML or text types were not included in the collection [61]. An assessment of the collections showed that WT10G does not exhibit a balanced bow-tie structure which is commonly observed in testbeds [117]. This may be caused by the processing that was performed in constructing WT10G. In contrast, the larger and unprocessed WT100G does have a balanced bow-tie structure.

<table>
<thead>
<tr>
<th>Description</th>
<th>WT10G</th>
<th>WT100G</th>
<th>WT10G/WT100G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of servers</td>
<td>11680</td>
<td>116102</td>
<td>10.06%</td>
</tr>
<tr>
<td>Number of domains</td>
<td>11672</td>
<td>115659</td>
<td>10.09%</td>
</tr>
<tr>
<td>Number of pages</td>
<td>1692096</td>
<td>18571671</td>
<td>9.11%</td>
</tr>
<tr>
<td>Number of html links</td>
<td>16123742</td>
<td>149681925</td>
<td>10.77%</td>
</tr>
</tbody>
</table>

As it can be observed from the basic statistical information contained in Table 3.2, the proportion of WT10G to WT100G is approximately 10% for the number of servers and domains, corresponding to the size ratio between the 2 collections. The number of pages in WT10G is lower than the expected 10%, which could be due to the focus on HTML type documents in the page extraction. On the other hand, the number of HTML links in WT10G is somewhat higher than the expected 10%; this could be caused by the intentional inclusion of pages with more inter-domain links.

A more detailed statistical description of the collections developed from empirical assessment is included in Table 3.3. The statistics on the number of HTML links in a page supports the assumption of inter-domain link inclusion, where there is a higher average link per page in WT10G than in WT100G. The statistics on the pages per domain indicate that during the construction of the WT10G collection, the domains of extreme sizes were filtered out.

Although care was taken to ensure that the WT10G collection represents the web data without containing undesirable features that would hinder experiments, several issues were identified in the collection. One of the major issues experienced during the data analysis process is the inconsistency of metadata. Some documents do not have the closing tag `⟨DOC⟩` to indicate the end of a web page. Also, various formats were used in the metatags at the document header. For
Table 3.3: A comparing of the statistical properties of the 2 TREC testbeds

<table>
<thead>
<tr>
<th>Description</th>
<th>WT10G</th>
<th>WT100G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average html links per page</td>
<td>9.53</td>
<td>7.89</td>
</tr>
<tr>
<td>Min html links in a page</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max html links in a page</td>
<td>14288</td>
<td>28594</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>66.27</td>
<td>37.82</td>
</tr>
<tr>
<td>Average pages per domain</td>
<td>144</td>
<td>160</td>
</tr>
<tr>
<td>Min. pages in a domain</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Max. pages in a domain</td>
<td>26505</td>
<td>43740</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>655.49</td>
<td>930.8</td>
</tr>
</tbody>
</table>

Example, the size of the html document could be displayed under “Content-length” or “Content-Length”, while some documents do not have information on the document size at all. Although there is a lack of standards for header information on the Internet, however, metatags inserted by the system could be standardised during or after the crawling of web pages.

Another issue is the incorrect documentation on the collection. The documentations about the collection are in the “info” folder of the distribution, containing information about servers and link relationships of each webpage in the collection. However, discrepancies between data and its documentation were identified. For example, in the “outlinks” file, document WTX001-B01-4 missed the out-link to document WTX001-B01-3. Another error is pointed out by Soboroff (2002), where when the “outlinks” file was transposed, the resulting set contains 109 links more than those documented in the “inlinks” file. This issue was recently addressed by the release of an update for the “inlinks” file.

All these extra processing procedures in WT10G resulted in a dataset that does not sufficiently reflect the Web properties. After the basic analysis, WT100G seems to provide a more accurate indication on the general characteristics of the Web.

Web page validity

The second major Web characteristic is the web page validity. From the basic properties of WT10G, it was obtained that there is a total of 1,692,096 web pages. However, during an empirical assessment conducted in September of 2004, only 5,884 of the 1,692,096 pages were still accessible on the Web, which is only 0.35%. Furthermore, the number of pages that are valid in the 2004 web was even lower than the number of accessible domains assessed at the same time-frame. This peculiar finding could be attributed to the large amount of change that has occurred.
in the 7 years since the construction of the WT10G testbed, where the same domain may still exist, but the pages may have different naming schemes or extensions, therefore can no longer be found.

Given the increasing use of dynamically created, user customized pages, it can be expected that the rate of changes in the Web will grow further. These points lead to the question whether a testbed that was developed from the web content 7 years ago is still useful. Although the WT10G and WT100G collections were representative of the web data in 1997, the collections seem to be outdated due to the dynamic nature of the web, thus unable to fulfill its initial goal of allowing experiment results to be applicable to the Internet.

Variation in Web content

The third major Web characteristic identified in the literature which is also apparent in the difference between the testbeds and the current web is the changing and increasing variety of web content. The popularity of the Internet has encouraged a wide variety of web content; therefore, there is a need to examine the amount of content change that occurred in web pages in order to determine whether the 1997 testbed collections continue to include an appropriate variety of subject areas. To evaluate the amount of subject change in web page content, an empirical assessment was conducted by analyzing the 5884 web pages from the WT10G collection which were still accessible in September 2004. This content evaluation assessment examined the change in content subject by extracting a vector of vocabularies from both the 1997 and 2004 versions of the web page residing in the same web address. Although 5884 pages were qualified, after the vocabulary extraction process, 141 pages with an empty vocabulary vector were removed. Then the dot product of the respective vocabulary vectors of the 1997 and 2004 versions is taken to yield a score that ranges from 0 to 1, where 0 indicates a total change of subject and 1 indicates no change.

From Figure 3.1, it can be observed that the web pages fall into 3 areas. From the 5743 valid pages, 1285 (22.38%) have a score of 0, which indicates that these pages have a totally different content in 2004. On the other hand, there are 798 (13.90%) pages with a score of 1, which remains the same even after the 7 year period. To further understand the changes that occurred in these webpages, the number of pages with a significant change (a dot product score of < 0.3) of content is identified. It is observed that more than two-thirds of pages have had a significant change in content, implying a change in the subject area of those web pages.

The observation that a large proportion of websites have undergone a significant change in content could be a result of the introduction of new subject areas over the years and the increasing
popularity of website maintenance tools which allowed information to be posted by a wide range of people. These new characteristics of the web are not reflected in the WT10G or WT100G collections, therefore confirming the need for a new yet comparable collection.

Variation in the file type

From the perspective of content file types, the change has been gradual, but especially problematic for information retrieval tasks. With the increasing adoption of server side programming and database utilization, there is significantly more dynamically-created web pages in the current Internet than 7 years ago. The absence of a physical location in dynamically constructed web pages entails the development of crawling agents to be more sophisticated and intelligent. Consequently, the data crawled from the Internet 7 years ago will not be able to reflect such a requirement. Therefore it can be concluded that the benchmark testbed WT10G and even its unfiltered WT100G collection are not suitable testbeds for experiments with application to the current web.

3.2.2 INEX 2005 movies collection

The XML Document Mining track provided by the INitiative for the Evaluation of XML Retrieval (INEX) in 2005 is a collection of XML formatted documents. The eXtensible Meta Language (XML) is a tool for describing many types of electronic documents. XML is related to HTML but is much more flexible in the naming of its definition and strict in its syntax. As such, XML is increasingly utilized to represent a wide range of electronic documents ranging from e-books, Web pages, to multi-media content. This is due to the fact that XML provides a flexible document format which allows cross-platform compatibility. Thus, documents created by an UNIX application can be viewed or edited on any other system with an XML capable application.
Data mining on such types of documents becomes of increasing importance as a consequence. The striking feature that makes data mining on XML documents so interesting is that XML provides a strict structural document description and hence, XML mining can predominantly be a structure mining task.

XML provides meta-tags which encapsulate content. These meta-tags can be nested and define the property of the content. A simple example, the following line of XML code: `<A>Hello <B>World</B></A>` gives a document which has the string “Hello World” as content where the word “World” shares the property `<A>` of the word “Hello” but also is being assigned the property `<B>`. E.g. “Hello” may be in italic font, and “World” may be in italic and bold font. It is not possible to have overlapping XML tags such as `<A><B></A></B>`. The consequence is that XML provides a tree-like decomposition of document content. In other words, any XML document can be represented by a tree-like graph structure.

This particular testbed consists of 9,640 XML formatted documents, each of which describes an individual movie, and contains attributes such as the movie title, list of actors, list of reviewers and many others. There is a total of 197 unique attributes found in this testbed. Each of the documents is labelled by one of the total 11 thematic categories and 11 possible structure categories.

Although XML follows a similar syntax to HTML, the difference is that the leniency introduced by browsers allows HTML documents on the Web to still be displayed even when they contain errors. Also, the relationship among documents through the use of hyperlinks in HTML documents cannot be found in this XML testbed. Therefore in these aspects, this testbed does not represent the documents on the Web well.

3.2.3 INEX 2006 journal paper collection

The XML Document Mining track provided by INEX in 2006 is similar to the movies collection provided in the previous year in the sense that the new testbed distributed in 2006 also consists of XML formatted documents. The difference in the 2006 testbed is that it comprises of more documents, and the clustering and classification problems are more complex when compared to the testbed released in the previous year. These changes result in the testbed being a better representation of real-world problems.

This testbed contains a total of 12,107 XML documents, and is divided so that approximately 50% is used for training purposes, and the other 50% for testing. Each of the documents in this testbed belongs to one of the 18 classes (journals), and the 18 classes do not have equal number of documents, therefore making this an unbalanced dataset.
Although efforts have been made to better represent real-world problems in designing the testbed, the documents in this testbed do not sufficiently reflect the Web documents, as it was not the intention. However, the use of such an XML collection as testbed has the advantage of containing labels, which allows experimental results to be evaluated.

### 3.3 Data collection instrument

It was found that the existing testbeds are either out-dated or do not reflect the documents on the Web, as they do not include spamming pages, pages generated by scripts, pages under new Top Level Domains, and other properties of Web data. Therefore showing that they do not sufficiently reflect the characteristics of the current Web. It was then decided that new snapshots of the Web would be taken to form new testbeds, and the new testbeds were created with the following intentions.

- To test the performance of the crawler
- To assess the amount of resources required for the crawling task
- To compare the new Web snapshots to previous benchmark snapshots
- To retrieve sufficient data for creating a new testbed for future experiments.
- To provide a closed system of Web documents for the development and testing of quality assessment procedures.

Note that the reference to crawler here serves two purposes. Firstly, a crawler for the efficiently retrieval of snapshots of a portion of the real Web, and secondly, a focused crawler for the quality information retrieval task. We will realize the development of these crawlers by firstly developing a scalable approach to crawling, then extend the crawler by a module to achieve targeted crawling for quality information.

We aim at crawling regular snapshots in order to assess the nature of change in the Web, and in order to have access to latest Web properties on which our methods can be developed and tested. As a result of this exercise, we have created one four snapshots so far and are in the process of crawling the 5-th snapshot. We are making these datasets freely available via the Web site http://www.artificial-neural.net/data/LeiDiCrawl/

For the collection of Web data to construct a more up-to-date testbed, a simple special purpose crawling application would be required. A crawler is an essential component of an information retrieval system. Therefore for the project, a crawler that is efficient, flexible, and features
sufficient functionalities to support the tasks to be carried out, is essential. Upon a preliminary investigation into the crawling applications currently available, it can be observed that many fundamental functionalities required for crawling the current web effectively are absent. In order to support those functionalities, the crawling application could need a total or partial restructuring; therefore, it was decided that a novel crawler needs to be implemented.

The implementation of a crawler for the project has three benefits: 1) It can be designed to better suit the tasks required for the project, 2) It allows a hands-on experience in dealing with web data, and 3) It assists the collection of web snap-shots at regular intervals, which could be used as the test-bed for experiments as well as allowing the observation of changes in the World Wide Web. The newly developed crawling application is arranged in a centralized distributed setting with particular focus on the crawling efficiency, flexibility and effectiveness. As a result of this exercise, we developed a crawler which is superior in some aspects to even the most elaborate crawlers. This will be shown later in this section.

One of the advantages of developing a crawler instead of using existing crawling applications is that the characteristics of the web can be better observed. This proved correct, as through the crawler implementation, the many challenges were able to be identified and then addressed. The issue of web browsers’ leniency for HTML syntax, and the issue of bad response from domain hosting servers are well known, but there are other issues that pose themselves as obstacles for web crawlers. There is currently no literature addressing the challenges of developing a crawler for the World Wide Web in a relatively comprehensive manner. Therefore, the description of development challenges that follow serves as a documentation of the characteristics of the web that are less commonly known.

- Non-terminating crawling loop

Symbolic links allow convenient access to directories in the hierarchy that are far apart, however, they can cause a crawler to crawl in a non-terminating loop. The following image illustrates the situation.

In a file structure similar to the image, where the dash line indicates symbolic link to a
physical directory, a crawling loop could occur. For example, if a page in dir1 contains a relative link to dir3, and that particular page in dir3 links to a page relatively via the symbolic link, it will result in the following, where the path names change but the files being crawled are essentially the same pages.

Table 3.4: Illustration of the effect of symbolic link on the link extraction process of crawling

<table>
<thead>
<tr>
<th>Physical location</th>
<th>File being crawled</th>
<th>Link in current page</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dir1/</td>
<td>/dir1/index.html</td>
<td>./dir2/dir3/index.html</td>
</tr>
</tbody>
</table>

Another situation that could lead to non-terminating crawling loop is when a web server is configured to redirect a web page without issuing a redirection response, in which case the redirection cannot be detected by the crawler. It is done by some web servers, for URLs that do not exist, so that those URLs without a valid web page are redirected to a default page that displays an error message. If the default web page that displays error messages has relative links, then non-terminating crawling loop would occur. This is because the crawler would not be aware that redirection occurred, and concatenate the relative link onto the URL that the crawler believes it is currently retrieving, resulting in yet another web page that does not exist, which would lead the crawler back to the same page. This cycle would endlessly repeat, causing a crawling loop. Such crawling loops are not easily detectable since a simple analysis for repeating patterns in a directory path can not distinguish between a recursive directory structure (i.e. caused by symbolic links) and valid directory with actual, physically existing directories with repeating names.

Crawling of a set of web pages in a continuous loop is a serious issue, as retrieving multiple copies of the same page wastes network resources that are not infinite nor free, and to continuously doing so in a loop additionally wastes the processing time and resource of one entire crawler. A possible solution is to restrict the directory depth, to prevent crawling in a non-terminating loop, however, this does not prevent the crawler retrieving multiple copies of a page. Another solution which the team adopted to address the issue is to detect symbolic links by comparing directory structure and content, to eliminate the possibility of
duplicated crawling. We observed that all major search engine providers opt for restricting the directory depth (mostly to a depth of 10). Hence, all of the popular search engines are currently unable to search information that is located deeper than the maximum depth in a directory structure.

- **File type misinterpretation**

  File extension and the content description in file header provide basic but important information about the content of a page, especially when efficiency and appropriate resource allocation are the major crawling concerns. For example, the "html" extension or file header content description of "html/text" indicates a static web page most likely to be constructed by ASCII characters, whereas the "exe" extension or the file header content description of executable indicates an executable file of binary content. These file type information help crawlers to determine whether a file is worth crawling, through the analysis of URL or file header before retrieval takes place. In recent years, as part of the dynamics of the Internet, leniency is introduced to web hosting servers in order to accommodate the rapidly increasing variety of file types. As a result, file extensions and file header description can no longer serve as an accurate indication for the file type, and therefore older versions of crawler would fail due to file type misinterpretation. A possible solution is to additionally examine the first few lines of the file to confirm the file type, which requires the crawlers to be able to manipulate string contents as well as binary contents.

- **DNS look-up time consumption**

  Due to the time and network traffic caused by DNS lookups, it is important to minimize DNS look-up to allow optimal crawling result. Preliminary test shows that the time taken for the retrieval of an IP from a database is approximately 0.00158 second, whereas the retrieval of an IP from the DNS server is approximately 1.66134 second. The difference is significant as retrieval from DNS server is approximately 1051 times longer than from a database, and the impact on crawling throughput will be serious during the actual execution where thousands of domains are being crawled. It follows then, that DNS look-up from various distributed processes is also not desirable. A possible solution, which was adopted in the research project, is to store the look-up results where all distributed processes could access, and only conduct a DNS look-up when a new domain is encountered.

- **Domains with multiple IP addresses**

  The dynamic property of the Internet allows several domain names to share the same IP, as well as a domain name having multiple IPs. This was initially done to allow flexibility
and to divert traffic so that servers are not over-loaded when there is high traffic, but this makes distributed crawling difficult, as a rule has to be set to avoid slaves that are crawling in parallel crawl the same data. The rule set in this distributed crawling application is that slaves have to distinguish individual sites, and crawl within the site, and the allocation of sites is managed by the master application, to avoid pages being repeatedly crawled by the slave or other slaves. The mechanism used to separate and identify sites, is to use unique IP addresses, which overcomes the popular practice of having multiple domain names sharing the same IP. However, a number of incidences were observed where domains with multiple IPs impact the effectiveness of the rule. This practice is sometimes used for larger domains to disperse the traffic accessing the website. A possible solution to this problem, which was adopted, is to maintain a domain and IP reference list where only one IP is assigned to a domain and no further DNS look-up is to be conducted.

The crawler is implemented in C and JAVA. The inter-process communication was initially managed through Parallel Virtual Machine (PVM), but was later converted to using a HTTP server due to the restriction on the accessibility of machines which participated in the crawling process. PVM was efficient and effective at handling communication between the crawling component and the central coordinator; however, it required access to a specific non-reserved port, which could only be opened with administrative permission, and that is not possible for some machines. The opening of a port also introduced some security risks which required addressing through additional security measures. Consequently, other options were considered, and the approach of utilizing a HTTP server to handle the inter-process communication was adopted for the crawling process. Details of each of the 3 crawling components will be provided in the following subsections.

The proposed crawler is utilized to retrieve snapshots of the web within a short time-frame, and for the development of up-to-date testbeds. The resulting crawling application is successful at performing such tasks, resulted in a publication at the Web Intelligence conference [71], and provides the following key advantages:

- Efficiency and decreased resource consumption
  - An intelligent slave management and seed distribution system keeps communication overhead at a minimum. This is achieved by giving slaves exclusive autonomy over the crawling of a site, and hence, seed distribution is greatly simplified.
  - It uses data compression and data encryption techniques to reduce the overall network use to a minimum. Slave processes compress crawled data before sending these to
the central data store. In other words, data compression is done in parallel by a
distributed system of crawling slaves, whereas the centralized server simply takes
the data and stores it in its local data store. The approach results in a very scalable
crawler.

- DNS caching and loop detection significantly reduce network overhead, redundancy,
and increase speed of crawling.

- The result is that the crawler is highly efficient for data retrieval, so much that delays
had to be inserted between crawling of subsequent pages so that the crawler will not
exhaust domain hosting servers with requests [71].

- **Flexibility**
  - It has flexible crawling arrangement, allowing slaves to join or remove themselves
  from crawling at any time without impacting the overall crawling task.
  - It runs on any operating system which may possibly be used for crawling tasks as
  long as such machines have the dependent applications (java) installed.
  - It uses a modular approach which allows it to bind with third party crawlers (ie.
    Special purpose crawlers for specific domains) with ease.

- **Novel features are introduced to support crawling effectiveness [71]**
  - It supports continuous crawling. When crawling is conducted on a site that has pre-
    viously been crawled, the crawler is able to only retrieve web pages that have not yet
    been crawled or that have been modified.
  - It recovers from faults and failures very efficiently. No data is lost in the case of
    a fault occurring, even if the fault occurs on the master node. This is achieved by
    maintaining a state for each slave and the master. The state is updated at regular
    intervals (i.e. 5 minutes), and hence, in case of a unexpected failure, the most that is
    lost (and may need to be re-done) is 5 minutes of work.
  - No maximum directory depth is assumed. This has become possible by the loop
detection module which ensures that snapshots of entire domains can be crawled
with minimal redundancy. This is the most significant contribution of our work on
this crawler. The mechanism will be explained in detail later in this section.
  - The partitioning rule is based on IP (Internet Protocol) addresses, therefore it is ca-
    pable of conducting truly localized crawling by working out the proximity of servers
    with respect to the crawling nodes through the analysis of IP addresses.
Figure 3.3: The structure and components of the proposed distributed crawler

Figure 3.3 shows the “master-slave” architecture of the proposed crawler. There are two components in the proposed architecture: the ”slave” which consists of a stand-alone crawling component, and the ”master” which acts as the central slave coordinating component, making this a centralized distributed crawling system. The following subsections will discuss the stand-alone crawling component and the central coordinating component in some more detail.

3.3.1 The stand-alone crawling component

The purpose of the crawling component is to crawl a portion of the web from a given set of starting (seed) pages as a stand-alone crawler. The stand-alone crawler is a depth-first crawler that retrieves data from the World Wide Web on a per domain basis. One of the major focuses in the design of the stand-alone crawling component is the communication with the central coordinating component. The initializing communication procedure between the crawling component and the coordinating program is as follows. The crawling components are initiated by a signal from the coordinating component. The crawling component then provides the configuration information of the computer it resides on to the coordinating component. The coordinating component will then provide a set of seed pages from within a domain which is physically located closest to the crawling host. Crawling would then commence immediately.

In a distributed setting, transfer of information to the central coordinating node may not be essential; however, to be comparable to and appear as though it is one crawling task from the central node, data should be transferred to the central node. The size of web content varies
greatly and consequently, the transfer of data will be in unpredictable bursts if transfer is to take place only at the end of each crawl. To improve the predictability of transfer and the stability of the coordinating program, throughout the crawling process, the crawled data is compressed, encrypted and sent to the coordinating component as soon as it reaches a predetermined size limit of 65KB. Moreover, the crawler can buffer crawled data in a local store until these can be sent to the central coordinator. In other words, the crawler processes are unaffected in cases of a communication failure with the master, or in the case of a master crash. The crawlers continue the crawl while re-trying to contact the master at regular intervals (every 300 seconds). Once the master becomes available, the accumulated data packages are moved to the master node. Upon completion of the crawl, a time-out of the server, or other errors, the scrawling components terminate following a procedure that closes all processes correctly and sends all left-over files to the coordinating component, together with the termination or error signal. This is implemented by adding a shut-down hook where a procedure is in place to make sure that the coordinating component is aware of the termination and that uncrawled URLs in the stack are saved onto the disk. The shutdown procedure ensures that the coordinating component of the crawling package will not wait for response from a crawling component that is not crawling for any reason, which allows the coordinating component to schedule a continuation of the interrupted crawl.

The other major focus in the crawling component design is the crawling restriction. Due to the dynamics of the World Wide Web, crawling restrictions are essential for minimizing crawling redundancy. The following restrictions are in place to avoid trapping in a crawling loop.

- The crawling component has the option to crawl within a site, where sites are separated by unique IP addresses.
- Loop detection and URL formatting to avoid infinite crawling loops.
- Links within comment tags or java-scripts are not followed by the crawler
- Dynamic pages that do not physically exist are not crawled to avoid crawling traps
- Redirected pages and pop-up windows are not crawled

Site based crawling is performed through the coordinating component allocating URLs that resolve to the same IP to a dedicated crawling component, the crawling component crawls within the domains that have the same IP address, and identifies URLs from other domains as external links. No additional DNS look-up is required from the crawling component. This ensures a clear partitioning between all crawling components, and supports local crawling.
As is illustrated in Figure 3.3, the stand alone crawler has five major components, and the details of each of the components are as follows.

1. The fate decider

The fate decider is the only module which is able to access the URL list directly to insert or obtain an URL. This independent crawling component is a multi-threaded application, where both the link extractor and the URL filter are in the form of threads so that multiple instances of them can be executed simultaneously. The advantage of increased efficiency from a multi-threaded crawling application also introduces potential synchronization issues. Therefore, restricting the access of the URL list to the fate decider module, which only has 1 instance during the program execution, ensures that the URL list is only accessible by at most one source at any given time. This prevents potential synchronization issues.

The procedure of tasks carried out by the fate decider is as follows.

The fate decider firstly obtains an URL from the URL list by contacting the URL list with an empty string. After obtaining the URL, the fate decider idles for a predetermined period of time, which is calculated based on the number of threads in execution at the time, before proceeding to the next task. This is done to ensure that the efficiency of the crawler does not overwhelm domain hosting servers and make the data retrieval process appear as a Denial-Of-Service attack, which affects other users’ access to the same resource or this application’s future access privileges.

Secondly, the fate decider initiates an instance of the HTTP input stream, and passes the obtained URL to it, so that the content of the particular URL can be retrieved from the Web. The fate decider has a time-out setting on the HTTP input stream in case the server that hosts the particular page traps the crawler, by keeping the application wait indefinitely for a response. The fate decider also keeps count of the failed retrieval attempts, and eight consecutive failures would imply that the hosting server may be down, therefore the domain would not be accessed for a period of time if this was the case. This mechanism was put in place to overcome the temporarily server downtime, which is commonly observed on the dynamic World Wide Web.

Thirdly, the fate decider creates an instance of the link extractor and passes the page content obtained from HTTP input stream to it. The fate decider is currently set to allow up to eight simultaneous link extractor threads, since the time requirement for link extraction is dependent on the size of the web document and the number of hyper-links. If link extractor
threads reach the maximum of eight, the fate decider pauses crawling until the number of
link extraction threads decrease.

Finally, there is also a URL filter, which filters the links identified by the link extractor
into a consistent format. URL filter is called for every hyperlink identified in the link
extractor, and is not invoked by the fate decider. Although this may result in URL filter
being called frequently, but the URL filtering process is expected to be fast. Once this is
carried out correctly, the resulting URL is sent back to the fate decider, which will analyze
the domain and decide whether to discard it, to store it as external links, or to add it to the
URL list. The URLs to be discarded are the ones already retrieved by the crawler, or the
ones in an ignore list. The URLs to be considered as external links are URLs which do
not match the partition rule. For example, links from a domain different to the seed URLs
will be treated as external links for domain-based partitioning. These external links will be
sent to the coordinating master node, so that another dedicated crawling slave may work
on them. The URLs from acceptable domains, and which are not already in the URL list,
or been crawled, will be inserted into the URL list by the fate decider.

Once this cycle is complete, the process is repeated by executing from the first step again,
until there is no URL left to crawl and that all link extractor are no longer active.

2. The HTTP input stream

The HTTP input stream retrieves the content of a given URL from the World Wide Web.
Several settings are in place to ensure error-free retrieval. Those include disabling redirec-
tion and pop-up pages, and restricting crawling depth in a loop dependent approach.

The prevention of redirected and pop-up pages is performed by disabling the “follow redi-
rection” option when establishing a connection to a web page. This prevents some unex-
plained idling of the crawler when accessing web pages with redirection or pop-up win-
dows.

Loop detection is a novel feature introduced to minimize the wastage of resources on dupli-
cated data, and to allow crawling to an unlimited directory depth, instead of the fixed-depth
crawling carried out by existing crawlers. Loop detection is useful for detecting recursive
structures created by symbolic links or mirrored sites. Loop dependent depth restriction is
achieved through the following procedure:

(a) Examine the full path of a link for repeated substrings.
(b) If a repeating substring is found, then compare the content of the page against a page
already retrieved which it is suspected to be a duplication of.

(c) If the contents of the two pages have 95% similarity, the page would be considered a
duplicated page.

(d) If 3 duplicated pages are found within a sub-directory, the directory will be consid-
ered a recursive directory, and the crawler will not crawl further into the affected
area.

Although it would be ideal to compare the contents of all web pages in order to identify
all occurrences of duplication, however, that would be extremely time-consuming for the
crawler to be rendered useful. Therefore, this approach of using the URL to provide a
clue to the potential of duplication, and then only comparing suspected pages maintains a
balance between accuracy and efficiency.

3. The link extractor The link extractor module searches through the content of a web page
for hyperlinks, however, links inside comment tags or within the body of java scripts are
not considered. The efficiency of the link extractor is dependent on the size of the web
document and the number of hyperlinks. When a hyperlink is identified, it is extracted and
passed to the URL filter to ensure a consistent standard-format string is obtained, and that
common malformation in the URL is corrected, before passing the URL to the crawling list
handler to be stored in the crawling list. It should be noted that the URL filter is invoked
only in this link extractor module, upon the discovery of a hyperlink in the web page,
which means that each hyperlink is processed individually, and that processing could be in
parallel to the processing of other hyperlinks.

4. The URL filter The URL filter is responsible for two tasks: URL formatting and filtering of
undesired web pages. URL formatting corrects the following common URL malformations
in the pathname, so that standard-format strings for path names are generated, for instance,
by:

- Removing the unnecessary indication of current directory ./
- Removing repeated // for root directory where one slash will suffice
- Enforcing a trailing slash to directory names
- Replacing special characters to a standard encoded format. For example, a space for
  %20, and a tab for %09.

The URL filter also has the functionality to filter out certain data types. For the crawling
tasks performed in the project, only HTML type files are crawled, as a result, retrieval of other data types is not permitted. However, this can be altered quite easily in the future.

Additionally, dynamic pages were filtered out, so as to avoid crawling automatically generated pages from a database. This is implemented because the automatically generated page (dynamic page) does not physically exist, instead, it misleads crawlers to believe that it is a new page which has not been encountered each time, even if the same page is being visited. This is achievable since some variables in the page-generating-script will be replaced by values in the database, making the URL as well as the page content appear different.

Another issue with dynamic page is that some web servers link a variety of documents to scripts, which a crawler would not be able to detect. A hostile web server could use this to mislead a crawler and deliver content different to what it seems to contain. This would place the entire information retrieval system in a great risk, therefore should be avoided.

Consequently, the crawling of dynamic pages is prevented by detecting URLs that contain parameters for the database, which often uses a combination of special characters such as the “=” or “&” sign, among others.

3.3.2 The central coordinating component

The purpose of the central coordinating component is to manage the multiple crawling components executing in parallel so that the crawling is performed in an organized fashion with minimum crawling redundancy and network overhead. During the design and implementation phase of the central coordinating component, four areas received the most attention, and they are the allocation of seeds, the handling of crawled data, the recycling of seeds and the updating of slave components.

1. Allocation of seeds

One of the concerns is the allocation of seed pages to the crawling components. Since the crawling components do not communicate with each other directly in a centralized distributed crawling, a procedure is required to ensure that the crawling components are aware of their responsibilities, and that no web page is crawled multiple times. Such a procedure is largely determined by the behaviour of the crawling component and its communication strategy through the coordinator. The following shows some methods that this can be managed.

- Seed only method
In the seed only method, the crawling components only crawl the given set of seed pages. No other URL is crawled. Newly discovered URLs (hyperlinks found in the retrieved Web pages that point to pages outside the set of seeds) are reported to the coordinator, and it is up to the coordinator to decide the number of URLs to include in each seed file, and which crawling component to allocate the seeds to. When a set of seeds are crawled, the coordinator will provide the crawling component with a new set of seeds. The advantage of this method is that the coordinator is well aware of the responsibilities of each crawling component, so the crawling redundancy can be avoided. However, since a URL needs to be sent for every page that is to be crawled, and hence, a high communication requirement results as the disadvantage.

- URL exchange method

In the URL exchange method, the crawling components will begin crawling outside a partition boundary once one has completed. The crawled and newly discovered URLs are reported to the coordinator, which passes the information to other crawling components. The seed file only needs to contain a small number of starting URLs, and only needs to be sent out once. The advantage for this method is that after the initial set of seeds, the crawling components is able to maintain their own set of seeds, instead of receiving them from the coordinator, therefore decreasing the communication overhead. The only information that is required from the coordinator are the identification of the crawled partition, and the newly discovered URLs from other crawling components. This method has the disadvantage of being more complex, and the amount of communication increases dramatically with increasing number of crawling components to avoid crawling redundancy by ensuring all crawling components are aware of what has been or is being crawled.

- Rule-based method

In rule-based method, the crawling components crawl strictly within a boundary. The entire data is partitioned according to a rule, which is implemented into the crawling components. The segment that each crawling component is responsible for, is decided by the coordinator as a crawling component becomes available. Therefore, the seed file sent to the crawling component only needs to contain a small number of starting URLs, and a new seed file needs to be sent after the responsible partition has been crawled. The advantage for this method is that it can avoid crawling redundancy and has the least communication overhead, because the coordinator can assume the entire partition to be crawled as soon as its responsible crawling component crawls;
therefore it is not necessary to report the crawled URLs back to the coordinator, instead, only newly discovered URLs outside the responsible domain or site need to be reported. The disadvantage for this method is that sometimes a web page within a partition is only discovered from other partition after the current partition has been crawled, therefore the coordinator needs to take this possibility into consideration, and be able to crawl web pages that were missed in an earlier crawl. For example, if the given rule is to “crawl all pages within a site starting from a given set of seeds” then the crawler would be expected to retrieve all web documents from a site that are reachable (through hyperlinks) from the set of seeds. It is possible that a site contains Web pages which are not reachable from a given set of seeds, and hence, these pages would not be retrieved. However, it is possible that such pages are reachable from a hyperlink which originated from a document on another site. By recursively applying the rule based method, and by including discovered remote hyperlinks to a site into the set of seeds, this ensures that all Web pages which are reachable by a hyperlink are crawed.

Since the crawling components do not communicate with each other, they would not be aware if another crawling component is crawling the same web page. Therefore, a partitioning rule is often useful to define the responsibilities of each crawling component so that an overlap in data crawled by crawling components can be minimized [30]. The following are a number of partitioning strategies [30].

- **URL based partition** - Each unique URL is treated individually, and may be grouped with URLs from other domain, site, or hierarchy. This partition strategy is usually associated with the seed only method.

- **Domain based partition** - The partition is defined by the domain name of URLs, so that web pages belonging to the same domain are grouped together.

- **Site based partition** - The partition is defined by unique IP addresses, which could include a number of domain names.

- **Hierarchy based partition** - The partition is based on the URL of a page. For example, partition could be done on the gTLD, such as one group for .com, one for .net, one for .edu and so on. The partition could also be done on the ccTLD, such as one group for .au, one for .de, and one for URLs without the ccTLD.

It was decided after analyzing the options to use the rule-based method, and to partition according to sites, as these options allow low communication overhead and only one set
of seed is required for web pages from the same web hosting machine. Often, low com-
communication overhead comes with a trade-off of high crawling redundancy. To assist in the
minimization of crawling redundancy, a database that contains domains, IP addresses and
other URL information, were utilized, so that URLs under the same site are grouped and
crawled by one crawling component only.

2. Handling of crawled data

The other concern of the coordinating component is the handling of crawled data. While
developing the crawler, a decision has to be made to whether keep the crawled data in
local machines, or to send them to a central location. Keeping crawled data locally will
dramatically decrease the network resources required for data transfer, but will be difficult
to manage as the crawling components may reside in machines with difference operating
systems with various amount of available disk space. Also, the data would eventually
need to be merged. Therefore, sending crawled data to a central location during crawling
appeared to be a better choice. However, a mechanism would need to be in place in order to
ensure that the data transfer process will not affect the main crawling task by requiring large
amount of network resources. The result was to split the crawled data into manageable
chunks of files, and to compress each file before data transfer takes place.

The coordinating component receives crawled data from the crawling component either
when the file size reaches a pre-determined maximum size, or when the crawling task for
a site has completed. All data received from the crawling components are in compressed
format, and the coordinator can decompress the data after successfully receiving it. The co-
ordinating component also keeps track of the last feed back time of crawling components,
so that if a crawling component has not signaled a completion, but has not responded for a
period of time, the coordinator will assume that the particular crawling component encoun-
tered an unexpected error such as power failure or some un-recoverable computer error. In
which case, the coordinator will traverse through the data received for the site up to that
point, to extract links so that external links and uncrawled seeds can be recorded. The site
will then be flagged as interrupted and be re-allocated to a crawling component at a later
time.

3. Recycling of seeds

The third concern of the master program is the recycling of seeds. As mentioned pre-
viously, a rule-based method was adopted to partition crawling responsibility according
to sites. To allow the site-based partition, the database requires information such as the
IP addresses, their corresponding domain names, and the URL of seeds among other information. An object-relational database management system (ORDBMS) called PostgreSQL was utilized to achieve this task. Due to the fact that the allocation of seeds are in accordance with the site-based partition, the crawling package has to accommodate the possibility of a new web page being discovered from another site after its site has been crawled.

To accommodate new web pages from a crawled site, the coordinator and the PostgreSQL database which the coordinator interfaces with, have to store some additional information. For example, the database stores domains, IP addresses, URLs and crawling status. Whenever a crawling component has completed crawling a site, the external links and uncrawled seeds (if any) from that site are returned to the master. The master program then inserts these URLs into the database. Interrupted sites are given a re-crawl opportunity, and will not be allocated for crawling again if both attempts fail. The dynamic nature of the World Wide Web means data on the web is constantly changing, so crawled data would have to be updated accordingly as well. Therefore, once all sites have been completely crawled or have had two crawling attempts, the database is reset and crawling restarts to ready itself for the creation of the next snapshot.

4. Information update

The fourth concern of the master program is the updating of information for the crawling components. This was implemented as a feature to allow more flexibility in the crawling application. Since it was decided that each site is allowed up to two crawling attempts and the crawling application supports continuous crawling by performing rounds of the crawling process. A mechanism therefore needs to be in place to inform the crawling components of the web pages already crawled, and the time of last retrieval, so that the crawling component only needs to retrieve a web page if it has not yet been crawled, or if the web page has been modified since it was last retrieved. This feature assists the task by updating each crawling component with information of crawled URLs and their last retrieval timestamp within the site that they are responsible for.

The coordinating component is kept as simple as possible without compromising on and robustness. This was done to limit the likelihood of having to deal with software bugs. However, as pointed out previously, in the extreme case where the coordinator encounters an error or fault which can be due to an unstable system, power outage, or acts of God, only the particular management instance is affected. Also, since the process handled by the coordinating component
are in short separate modules, therefore, will not block other instances from their tasks. If the
server is down temporarily, the crawling tasks carried out by crawling components would not be
affected. Only the transfer process from crawling components will be affected in that it will be
suspended until the server and the network are back up.

3.3.3 Communication strategy

The distributed platform is based on server technology, which is flexible and secure. The decision
to adopt this communication strategy is to ensure secure, platform independent communication
while offering flexibility and reducing cost significantly. All data transfer between the master
and slaves are compressed to minimize network resources. The actual communication procedure
is as follows.

1. Crawling component signals the coordinator when it is ready to crawl.

2. Once connection is established, the slave provides configuration information about the
machine it resides on to the master. Amongst other reasons, this is to allow the master to
assign seed pages which are located closest to the slave.

3. Coordinator then sends a compressed package, which contains a set of seed pages, and in
the case of a recrawl, a list of crawled pages including their time stamp.

4. Slave commences crawling immediately after receiving the seeds.

5. During crawling, crawled data is compressed and sent to the master when it reaches a
predetermines packet size limit.

6. When crawling completes, the last crawled data file and other information files are com-
pressed, and then sent to the master.

7. When master receives the information files, it realizes that crawling has completed for a
site, so the cycle is repeated from step 3.

In the case where the crawling component transfers data to the coordinator while the coordi-
nator is engaged in processing other crawling component’s data, the new data would be handled
by a new instance of the management process, which would execute in parallel, and the crawling
component would not be required to wait. In the scenario where packets are lost or other errors
occur to cause a failed or incorrect transfer, the transfer will be repeated and data on the slave
node is only removed after a confirmed successful transfer, which ensures that no data will be
lost.
In another case where connection between the master and slave nodes cannot be established, the slave node continues with its crawling task, and stores the crawled data on disk temporarily. Another attempt to establish a connection for data transfer is conducted every 300 seconds until successful, upon which all data stored on disk are transferred one after another. If crawling completed while a connection cannot be established the slave node does nothing until connection is established and data transfer can take place. After all data are transferred to the master node, a new set of seeds will be sent to the slave node, and a new crawling task for a different site can begin.

3.3.4 Realization of design goals

As mentioned earlier, this crawling application was designed with 3 goals in mind: efficiency, flexibility and effectiveness. This section will describe how these goals are achieved, and the implications of these goals.

Crawling efficiency

In order to achieve this goal, a number of features were implemented into the crawling application with the specific aim of minimizing the network usage, communication overhead and crawling simplicity. The decision for a distributed crawling application is to allow an increased retrieval throughput by increasing the number of simultaneous crawling slaves. DNS caching, and loop detection significantly reduce network overhead, redundancy, and increase retrieval speed. A minimized communication overhead is achieved through data compression for the communication or data transfer between the master and slave components. Crawling simplicity is ensured by keeping the application codes down at about 2000 lines of code, which reduces maintenance cost considerably, and allows to write the code bug-free.

Application flexibility

The JAVA based crawling slave offers flexibility with its ability to run on any operating system which may possibly be used for crawling tasks. This goals is further achieved through incorporating various option in the crawling application.

- Flexibility in the crawling boundary - Restrictions can be by site, domain, or specific URL
- Flexibility in the types of data to retrieve - Can retrieve HTML and text based document, only XML documents, only multimedia documents, or all documents including binary data. A variety of combinations is possible as well.
• Flexibility in the data storage method - Users can choose to store the crawled data on disk, to only keep track of crawling statistics, or to not store any information.

Crawling effectiveness

No maximum directory depth is assumed. This has become possible by the loop detection module which ensures that snapshots of entire domains can be crawled. This offers a different perspective on how the Web data is perceived, as a more complete view of domains can be obtained, instead of being bounded by a fixed depth of file structure.

As indicated in Chapter 2, popular search engines have access to a portion of the Internet. This is called the visibility problem of the Internet [136]. It is known that the visible web, especially for small search exercises, like those used in personalized search engines, depends on the seed pages: the initial pages which are used to start off the search process. But as the crawler crawls more and more of the Internet, the influence of the seed pages on the visible web would become less and less. One way to minimise the effect of the influence of the seed pages on the visible web is to use the top level domain directory list which provides a list of all the major domain directories in the existing web (please see article on generic top level domains in [135].

3.3.5 Experiments

During the development of the crawler, several experiments were conducted to observe the significance of a feature and to examine the behavior of the crawler. In particular, experiments were conducted to observe the amount of improvement in crawling efficiency under a distributed setting, and the degree of crawling effectiveness with using content based depth restriction.

Improvement in efficiency under a distributed setting

It is commonly believed that a distributed system is more efficient than a single sequentially executed crawling process because of the fact that a distributed system allows multiple processes to run simultaneously over a number of locations, therefore the same task can be achieved in much less time. However, a distributed system also has drawbacks. The management of all the individual processes and the communication between the processes may result in a complex system that requires so many additional tasks that it is no longer efficient. This leads to the question: How much faster is a distributed system over the single sequential process in practise?

The distribution tests carried out aim to identify the impact of the management and communication requirement in a distributed system, as well as to compare the throughput achievable in
Table 3.5: The average throughput achieved by crawlers in various locations with various crawling approaches

<table>
<thead>
<tr>
<th>Machine ID</th>
<th>Throughput without delay</th>
<th>Throughput of sequential crawl</th>
<th>Average throughput of distributed crawl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local 1</td>
<td>0.668 sec/pg</td>
<td>2.515 sec/pg</td>
<td>2.564 sec/pg</td>
</tr>
<tr>
<td>Local 2</td>
<td>0.668 sec/pg</td>
<td>2.498 sec/pg</td>
<td>2.754 sec/pg</td>
</tr>
<tr>
<td>Domestic 1</td>
<td>1.191 sec/pg</td>
<td>2.835 sec/pg</td>
<td>2.834 sec/pg</td>
</tr>
<tr>
<td>Domestic 2</td>
<td>1.112 sec/pg</td>
<td>2.694 sec/pg</td>
<td>2.524 sec/pg</td>
</tr>
<tr>
<td>International 1</td>
<td>1.598 sec/pg</td>
<td>3.100 sec/pg</td>
<td>2.934 sec/pg</td>
</tr>
<tr>
<td>International 2</td>
<td>1.860 sec/pg</td>
<td>3.396 sec/pg</td>
<td>3.291 sec/pg</td>
</tr>
</tbody>
</table>

various distributed network settings. In the distributed tests carried out, a centralised distributed system is adopted.

A.) Impact of additional tasks in a distributed system. In order to compare the throughput of distributed crawling and the basic sequential crawling, the factors that could have an effect on the throughput should be identified. For the distributed crawling, those factors include communication and the transfer of data to a central server. For the sequential crawling, the factor is the amount of data that the internal stacks have to keep track during crawling, which makes seed insertion and retrieval more time consuming.

In addition, the crawling application has a feature to avoid overloading crawling domains with requests, which means, there are delays in between the crawling of each page, and this would affect the crawling throughput for both the test using a distributed system and the test using a single sequential crawler. Table 3.5 shows the throughput of the various scenarios, where the statistics from the distributed crawl was obtained using 2 parallel crawling processes. It can be seen that the main advantage in speed is gained through the support of local crawls (from a nearby site). This can only be achieved with a distributed system since a sequential crawler is restricted to reside on one location only. Moreover, it is observed that a distributed system with two crawlers retrieve pages at about the same speed as a sequential crawler. This shows that the overhead of the distributed system has a negligible effect on the average retrieval speed of Web documents. Not that the right column in Table 3.5 refers to the average retrieval speed per page. But since there are multiple (i.e. $n$) crawler processes running, and hence, the overall time for crawling the same set of pages is reduced by the factor $\approx n^{-1}$. 

83
There are 2 major areas for discussion from this portion of the test: the throughput difference with different method of crawling, and the throughput difference depending on the geographical location of machines.

1. Method of crawling

Firstly, there is a need to take a look at the effect of the delay feature in the crawler. The delay between crawling each webpage is dependant on the number of existing processing thread. The crawling application allows up to 8 threads to process the crawled data and extract the links, so the delay is set to extend with more number of active processing threads, to prevent the system from being too busy. The effect of this, as can be observed from the table above, is that the crawlers with the delay feature are slower by about 1.5 seconds per page, except when the crawling is done on a local domain. Therefore, instead of the usual case where network is the bottleneck, the processing time becomes the bottleneck when crawling in the fast local network. Since both sequential crawling and distributed crawling have the delay feature, the throughput can be compared without any problem. It can be seen from the table above, that sequential crawling seems to perform better when the speed of the network is fast, such as when crawling a local domain. Otherwise, distributed crawling has a faster throughput in average.

2. Geographical location of machines

As Table 3.5 had shown, the throughput differences among local, domestic and internationally located machines are quite significant. Using the sequential crawling method, these locality issues would not be important, since there is only a single point of crawling, so most of domains would be crawled at the international speed. In the contrary, the distributed crawling method will be able to take advantage of this difference in throughput between various localities. A distributed system will be able to allocate the domain to the nearest machine, which could increase the efficiency of the crawling system dramatically.

The finding from this portion of the test shows that the throughput between using the sequential and distributed crawling methods are similar, sequential crawling is more efficient in some cases, and distributed crawling is better in other cases. However, bear in mind that distributed crawling uses multiple crawling processes at the same time and can take advantage of machines’ locality; therefore distributed crawling is the more efficient crawling method.
Table 3.6: The crawling throughput achievable with 1 to 4 simultaneous processes

<table>
<thead>
<tr>
<th>Locality</th>
<th>1 process</th>
<th>2 processes</th>
<th>3 processes</th>
<th>4 processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td>2.564 sec/pg</td>
<td>2.549 sec/pg</td>
<td>2.571 sec/pg</td>
<td>2.578 sec/pg</td>
</tr>
<tr>
<td>Domestic</td>
<td>2.834 sec/pg</td>
<td>2.761 sec/pg</td>
<td>2.800 sec/pg</td>
<td>2.848 sec/pg</td>
</tr>
<tr>
<td>International</td>
<td>2.934 sec/pg</td>
<td>3.086 sec/pg</td>
<td>3.116 sec/pg</td>
<td>3.131 sec/pg</td>
</tr>
</tbody>
</table>

B. Achievable throughput in various network settings. Now that it has been established that a suitable realization of distributed crawling is more efficient, this portion of the test explores the crawling performance under various settings in distributed crawling.

1. Investigate the impact of crawling process on the resources.

In this test, a domain is crawled multiple times, each time with an added crawling process, to investigate the impact on local resources such as CPU and network bandwidth, and the impact of the crawling performance when those resources are being shared. The machine deployed as a crawling slave for this experiment is equipped with 3.2GHz Intel Pentium 4 CPU, with 512 RAM. Its maximum network throughput achievable is limited to 100Mb/s.

We observed that the CPU usage for 1 stand-alone crawling process on the machine deployed as a crawling slave ranges from 5% to 80%, but with a low average of 15%.

It was observed that the CPU usage remains relatively low for most of the crawling duration, but spikes once in a while, and for a very short duration each time. The reason for the CPU usage to reach as high as 80% is due to the crawling process compressing the crawled data when it reaches the pre-defined packet size. Because of the CPU resources required for data compression, a slight decrease in performance is expected even with just 2 crawling processes. Network-wise, a decrease in performance is only expected when 3 or more processes are retrieving from the web at the same time.

In practice, as can be seen from the Table 3.6, the difference in throughput between 1 and 2 processes is greater than that between 2 and 3, which is also greater than the difference between having 3 and 4 processes. This is perhaps because when a crawling task is split into groups, the machine will only be busy for a short period initially, then when the shorter processes finish, the longest running process will be able to crawl at its usual speed. The more processes used, the more groups that the task was divided into, and the smaller each group will be; therefore, when more
2.4
2.6
2.8
3
3.2
3.4
3.6
3.8
4
1  2  3  4  5  6  7  8  9  10
Throughput (seconds per page)
Number of simultaneous crawlers

Figure 3.4: Illustration of the crawling throughput achieved when increasing number of crawlers are adopted in parallel

As Figure 3.4 shows, in the best case where network of the hosting domain is not busy from other users, there is no significant decrease in crawling efficiency. There is only a slight decrease when the number of simultaneous crawlers reaches 9 and 10. However, in the worst case where the network is already busy with other users, the throughput decreases quite dramatically with an increase of simultaneous crawler. Therefore, the impact of the crawler on the hosting domain is heavily dependant on the network condition at the time.

Furthermore, in the graph, a trend of gradual decrease of the average throughput seems to form with the increasing number of crawlers, and this decrease is only more
apparent for 8 or more crawlers. Considering the scale of the throughput in the chart, this decrease in the average throughput is not significant, and shows that even with 10 crawlers all accessing the hosting domain simultaneously, would not cause any network congestion.

3. Examine the change in throughput during a crawl

The speed at which the data are being retrieved may vary during a crawl. It would be useful to observe the amount and area of change in the throughput, to further understand the nature of the crawling application.

Figure 3.5 show the throughput during the progress of a crawling process. The 2 lines illustrate the crawling throughput from a local machine and from a machine located in a different country to the hosting domain. From the chart, it can be seen that the variation of throughput during the crawling process is similar for both the local and international crawling. They both show a slow start, but then quickly drop down to a stable throughput, and remain stable with slight variation until the completion of the crawl. There is no sign that the throughput decreases as time progresses, therefore it can be inferred that the efficiency of the crawling application will not be compromised with crawling of a large domain over a long period of time.

**Improvement in crawling effectiveness with content based depth restriction**

Most crawlers use a fixed-depth method of crawling, where depth is set to 10 or 11 (eg. Google’s crawling agent). A new approach that uses URL information and page content, to provide clues to whether the webpage belongs to a recursive directory, and restrict the depth accordingly, appears
to be more effective.

One may argue that the amount of unnecessary data obtained through crawling recursive directories, up to a fixed depth, is insignificant. However, in a dataset crawled in December 2005 using a fixed-depth approach where depth restriction was set to 10, 8.95% of crawled pages was unnecessarily crawled due to recursive structures in various domains.

The loop detection feature in our Lightweight Distributed (LeiDi) Crawler is a novel and important addition that does not exist in other known crawlers. The loop detection method equals or outperforms its fixed-depth counterpart in 3 major aspects: efficiency, redundancy reduction and retrieval coverage.

1. Efficiency

The impact of our loop detection algorithm is an average 2 milliseconds per page. This shows that although loop detection seems to involve more complex tests, the impact on throughput is not of a significant impact.

2. Redundancy reduction

Page redirections, domain mirroring, symbolic links within the domain structure can all result in redundant web pages. The amount of resource saved by avoiding crawling duplicated web pages is dependent on the directory structure of each domain. In a test conducted on www.crown.net - a domain with hidden redirections demonstrated the extent of redundancy reduction that could be achieved using the loop detection feature in crawling. During the test, crawling was conducted twice, once with the fixed-depth set to 10, another with the loop detection feature. The fixed depth crawling took a little over 34 hours, crawling 26442 pages, whereas the loop detection crawling only took about 8 hours to crawl 8180 pages, eliminating the need to crawl 18262 redundant web pages in that domain. Therefore, the reduction that can be obtained using the loop detection is evidently significant.

3. Retrieval coverage

The problem with having a fixed-depth restriction to URL is that some pages positioned deep in the structure may be missed by the crawler. Therefore, a test was carried-out to observe the type of information located deep in the domain structure and whether it is worth crawling in deep structure.

We found that not many domains use directory structures deeper than 10. As an example, the domain www.popularmechanics.com was discovered to use deep structure extensively. In the crawling test, 37,165 webpages were crawled, and 24,775 of these pages have URLs
with a depth of more than 10. This finding was surprising, as this indicates that the traditional fixed-depth crawlers would have missed all these genuine webpages that are deeper in the structure. This can be easily verified by conducting a search on Google for files on that domain. It is found that Google indexes the domain but does not index the 24,775 pages located deep in the directory structure. Hence, the majority of web pages in that domain are not visible in Google. The observation remains even after changing to any of the other popular search engine such as Yahoo! or MSN search.

The crawler implemented for the research project used a basic breadth-first crawler as a basis with options on the crawling approach and types of data to retrieve. The basic breadth-first crawler is comparable to other crawling applications in terms of throughput, however, the crawler also boasts some features that have not been explored previously, such as truly localized crawling and loop detection which allowed a content-dependent depth restriction. As the experimental results show, these two features significantly improves the efficiency and completeness of crawling by minimizing wastage of resources on international data retrieval and on redundant crawling. These approaches are novel, and as a result, allowed for a different perspective of the Web.

3.4 Properties of the snapshots

As part of the evaluation and test routine during the development of the crawler, we have crawled several snapshots of portions of the web. The snapshots were taken at 3 months interval. Snapshot 1 was taken in June of 2005, snapshot 2 in September 2005, and snapshot 3 taken in December 2005. The latest snapshot contains a total of 35,501,425 pages from 8,623 different domains, located across 1,853 different servers. All types of domains were considered including commercial, educational, government, non-profit organisational non-English, etc. domains. When analyzing the crawled data for statistical properties and when preparing it for use as testbeds, the non-English, duplicated and dynamically generated web pages were removed from each of the snapshots. This assists the construction of testbeds, on which some observations were made.

3.4.1 Basic properties

As mentioned earlier, the retrieval of the snapshots were carried out in 2005 with 3 months interval. From each of the snapshots, 2 datasets were created: one being the un-processed dataset, which retains the properties of the web snapshot, and the other is a filtered dataset which had the non-English, duplicated and dynamically generated web pages removed for better suitability as a
test-bed for the information retrieval experiments to be carried out as part of this research project. The basic statistical properties for the snapshots are summarized in Table 3.7.

<table>
<thead>
<tr>
<th>Table 3.7: The basic statistics of the retrieved web snapshots</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BEFORE FILTERING</strong></td>
</tr>
<tr>
<td>Number of pages retrieved</td>
</tr>
<tr>
<td>Number of domains accessed</td>
</tr>
<tr>
<td>Number of servers</td>
</tr>
<tr>
<td><strong>AFTER FILTERING</strong></td>
</tr>
<tr>
<td>Number of pages</td>
</tr>
<tr>
<td>Number of domains</td>
</tr>
<tr>
<td>Number of servers</td>
</tr>
<tr>
<td>Total uncompressed size</td>
</tr>
</tbody>
</table>

From the basic properties displayed in Table 3.7, it can be seen that although the snapshots were taken at regular intervals, they are of varying sizes. All 3 snapshots used the same set of seed pages, which are the page URLs from the WT10G testbed and their hyperlinks. The snapshots were all collected within a time-frame of approximately 4 weeks. The variation in the set-up is the number of participating crawling nodes, which increases and decreases a number of times during the crawling process of each snapshot collection. Snapshot 3 was especially collected with a larger number of crawling nodes. More specifically, an increase of 2 crawling nodes in average were used during the collection of snapshot 3, compared with the previous 2 snapshots.

The sharp increase in the number of Web pages retrieved between the snapshots, using the same set of initial seed pages, is due to two main factors: (1) it reflects the increase of the size of the Web, and (2) the realization of new features into the crawler such as loop detection which allowed us to exhaustively retrieve all reachable pages from a given set of domains, and due to improved robustness of the crawler which became able to deal with faults and errors much more effectively. The decreasing domains between the snapshots is due to the removal of many domains. We initially started with a set of 24,339 domains. Only 8,623 of these domains remains when conducting snapshot 3. Up to that time we have not considered adding new domains to the dataset. The increase in the number of servers is perhaps due to the change in the trend of virtual hosting. It was observed in snapshot 1 that some servers use name-based virtual hosting extensively, and as a result, up to 3822 domains would resolve to the same IP address, which is used to indicate unique servers. In the later snapshots, the maximum and the average number of domains associated with an IP address have both decreased, therefore showing that less domains
are now being crawled from more unique servers.

Further analysis on the testbeds allowed the compilation of statistics beyond the basic properties, as the investigation into the change in domains located on a server has already unveiled an interesting trend in virtual hosting. Other detailed statistics about the snapshots are shown in Table 3.8. It should be noted that the more detailed statistical information contained in Table 3.8 is derived from the filtered dataset, where the filtered snapshot 2 has less number of web pages than in the filtered snapshot 1, and the filtered snapshot 3 is dramatically larger than both of the previous snapshots. These can be seen in the lower segment of Table 3.7. In the more detailed Table 3.8, it shows that although the maximum size of web pages seems to increase from approximately 3,763,078KB in snapshot 1 to approximately 4,157,801KB in snapshot 3; however, the average size seem to be decreasing. This suggests that although the maximum file size of web pages is increasing, there is an increasing number of web pages with a small file size to shift the average lower in the more recent snapshots. The links per page detail shows an increasing number of hyperlinks and an increasing variation in the number of hyperlinks in web pages as well.
3.4.2 Implications of the statistics

The compiled statistics from the previous subsection can be very useful. For example, there are many attempts to estimate the size of the current web; with the knowledge of general statistics, an up-to-date estimation on the size of the web could be projected.

One of the most recent estimates, which is based on analysing pages indexed by various search engines, indicated that there were more than 11.5 billion indexed web pages in 2005 [51], and this grew to at least 17 billion by February 2007 [38]. An estimate of the publicly accessible portion of the Internet claims that there were 108.81 million web sites in February 2007 [94]. Given that the average size of a domain crawled during our latest crawls was 1157.97 pages, we can interpolate a possibly more accurate figure of 126.00 billion web pages \(^1\) on the web as of February 2007. The average size of a domain is calculated from the unprocessed web snapshots, using the crawler which is specifically well-suited for crawling entire domains, but is able to avoid crawling duplicated data. Crawling was conducted on a total of 44,008 domains, and the calculated average number of web pages from these domains is then multiplied by the total number of domains reported for February 2007. This estimation is made based on the knowledge of a large number of domains which were retrieved as completely as possible, and the snapshots were retrieved at different periods within a year. Also, no processing was involved to alter the proportions of the domain size. Therefore, although the estimation is significantly larger than the estimate of the size of the web made by others, this estimation can be considered valid. This would then imply that the web is growing at a faster rate than most people expected.

There is a caveat in this estimate: it is based on the average size of domain crawled by our crawler. The size of the web estimated in this manner is sensitive to the average size of the domain. Although the estimation on the size of the web based on the average number of pages per domain as observed through the newest testbed, does not agree with the estimation based on other approaches, but further investigation which involves other statistics may reveal the most accurate size estimation. However, the size of the web is not the only web property of interest. Perhaps other web properties can also be estimated using the statistical information of the testbeds as a basis. More importantly, it the identification of trend, so that a projected estimation of the future web can be made.

\(^1\)We only count physically existing HTML pages.
3.5 Trends in Web dynamics

Understanding the approximate size and the general statistics of the World Wide Web provides an awareness of the amount of process required by applications designed for the web; however, a more useful approach would be to observe the type and amount of change, to identify trends, which may allow the prediction of possible future changes on the web. The observation and identification of trends in the way in which the web is changing, is made possible through the analysis of 3 snapshots which were collected during the course of the research project. Basic statistical information about the snapshots can be found in Section 3.4, and observations of trends from the snapshots are reported in this section.

3.5.1 The rate of page inaccessibility

The number of web pages on the web can be influenced by two factors: the rate at which web pages become inaccessible, and the rate at which new web pages are generated. It has been widely reported that the number of people who has access to the Internet and who has their own homepage is constantly increasing [11, 65, 58]. However, this does not imply that the web is increasing at the same rate. There are many factors that could decrease the growth of the web, such as the server being temporarily unavailable, the deletion of web pages and the restriction of access to web pages. This subsection will examine the rate at which web pages become inaccessible, in order to more accurately predict the size of the Web.

The Online Computer Library Center conducted a routine analysis of the Internet from 1998 to 2002, which revealed some significant trends in the way that the Internet is changing. One of the observations is the decrease in the proportion of the publicly accessible Internet and the gradual increase in the proportion of the private web (intranet), where a private web is defined as websites where the content is intended for a restricted audience, controlled in the form of authorization or membership requirement [100].

Another observation is the rate of website inaccessibility, consistently at approximately 50% per year. The observation on the evolution of the Internet is extended to web pages, where the inaccessibility is at the extreme rate of 80% per year [97]. Investigation was conducted in 2004 by our research team [28], comparing the predicted rate of change on the web as observed by [97] to the empirical findings. The investigation showed that the assumptions of existing models about the dynamics of the Internet are inaccurate, and that the rate of change in the Internet is at a slower acceleration than predicted in the literature [97, 100]. This discrepancy between prediction of change on the web from the literature and empirical evaluation is also observed by
The empirical study by [97] demonstrated a high degree of change in the Internet. They stated that approximately 50% of existing domains will no longer be accessible after one year, and the rate that web pages become inaccessible is at 20% each year. To verify this, the domains and web pages from the WT10G collection were assessed for their accessibility in the web at the time of assessment (2006), by establishing a connection with the corresponding URLs. The accessibility was determined by analysing the response codes received from the connections. During the assessment, some problems were encountered, such as page redirection, automatic generated domain reselling pages and non-html pages. The domains or pages with the mentioned circumstances were not considered valid. The analysis results is summarized in Table 3.9.

Table 3.9: Accessibility of pages and domains in the WT10G collection. Valid as of September 2004

<table>
<thead>
<tr>
<th></th>
<th>Domains</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number in 1997</td>
<td>11,672</td>
<td>1,692,096</td>
</tr>
<tr>
<td>Expected accessibility in 2004</td>
<td>91 (0.781%)</td>
<td>17 (0.001%)</td>
</tr>
<tr>
<td>Actual accessibility in 2004</td>
<td>5984 (51.27%)</td>
<td>5884 (0.35%)</td>
</tr>
</tbody>
</table>

As Table 3.9 shows, of the 11672 domains in WT10G, 5984 (51%) were still valid in September 2004. The value in the expected accessibility column is derived from the study in [97], which observed that approximately 50% of domains are not accessible after one year. Based on the assumption of a linear change in web contents, as observed in the experiment by the Online Computer Library Center for the period of 1998 to 2002 [97], a compound rate method is used to project that a collection from 1997 should have approximately 0.781% of the existing domains accessible in 2004.

From Table 3.9, of the 1,692,096 pages in WT10G, only 5,884 (0.35%) pages were valid in September 2004. The expected accessibility identified in [97], where web pages were observed to become inaccessible at a rate of 80% per year, imply an accessible rate of 0.001% after 7 years. However, our empirical assessment does not agree with the rate predicted in [97]. More specifically, our empirical findings from observing the web data between 1998 and 2004 revealed that, instead of the predicted inaccessible rates of 50% and 80% per year for domain and web pages respectively, the inaccessible rates are only at 9.10% and 55.42%. Indicating that the web is changing less rapidly than expected. Another observation from more recent snapshots, naming the three snapshots obtained in 2006, also revealed a similar inaccessible rate, which is not as high as those stated in the literature. Therefore, confirming that the web is changing less rapidly.
than expected, because the rate of web page inaccessibility observed is lower than reported in literature.

### 3.5.2 The variation in file extension usage

After the investigation into the factors that influence the number of web documents, an interesting and increasingly noticeable phenomenon observed during crawling should also be explained; that is, the phenomenon of a rapidly expanding number of file extensions. Similar to the size of the web, the number of file extensions found on the web is also increasing at an amazing rate. There were 2619 different file extensions found in the hyperlinks within web pages in the 1997 benchmark dateset - WT100G; by 2005, the number of different file extensions found in a comparison web snapshot has increased to 15,349. Although some of the file extensions can be created or manipulated without having to correspond to the particular type of file that the extension signifies. However, in most cases, the file extensions are correctly used, because it allows the correct assignment of application to interpret the file content. Therefore, this large number of unique file extensions provides a clue to the number of varied data format in files accessible from the web.

This section will additionally look at the changes in the popularity of file extensions other than “html” and ‘htm”, through the analysis of hyperlinks. Table 3.10 shows the top five commonly found file extensions, which were obtained by analyzing the hyperlinks of web pages from the three snapshots. The order of the file extensions in Table 3.10 seems to vary among the snapshots, however, some file extensions are in the top five consistently. They include “shtml”, “php” and “asp”, two of which represent pages that can be dynamically generated, and the other one “shtml” is for a HTML page which has security restriction. This corresponds to the observation noted by [100], which states that there is a gradual increase in the proportion of the web for a restricted audience, controlled through authorization or membership requirement. It should be noted that “html” and “htm” file extensions are by far, the most commonly found in hyperlinks;

<table>
<thead>
<tr>
<th></th>
<th>Snapshot 1</th>
<th>Snapshot 2</th>
<th>Snapshot 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>most popular</td>
<td>shtml</td>
<td>php</td>
<td>shtml</td>
</tr>
<tr>
<td>2nd</td>
<td>zsp</td>
<td>m3u</td>
<td>php</td>
</tr>
<tr>
<td>3rd</td>
<td>asp</td>
<td>jsp</td>
<td>cgi</td>
</tr>
<tr>
<td>4th</td>
<td>php</td>
<td>shtml</td>
<td>asp</td>
</tr>
<tr>
<td>5th</td>
<td>jhtml</td>
<td>asp</td>
<td>gif</td>
</tr>
</tbody>
</table>
however, they are not listed in both of the tables so that other file extensions can be investigated.

There is a possibility that a web page with many hyperlinks to files of the same extension will easily affect the order of the file extension popularity, therefore, another analysis was conducted. In the second analysis, only unique file extensions in each web page are recorded. Using this approach, multiple hyperlinks to files of the same extension within the same page will not distort the order of popularity. Also, the percentage of web pages containing a link to files of certain extension can be calculated, as the maximum number for any file extension will be the total number of web pages analyzed. Table 3.11 shows the file extension popularity using the new approach.

Table 3.11: The top 5 popular unique file extensions per page in the web snapshots

<table>
<thead>
<tr>
<th></th>
<th>Snapshot 1</th>
<th>Snapshot 2</th>
<th>Snapshot 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most popular</td>
<td>asp</td>
<td>php</td>
<td>php</td>
</tr>
<tr>
<td>2nd</td>
<td>php</td>
<td>asp</td>
<td>asp</td>
</tr>
<tr>
<td>3rd</td>
<td>shtml</td>
<td>jsp</td>
<td>cgi</td>
</tr>
<tr>
<td>4th</td>
<td>cgi</td>
<td>aspx</td>
<td>shtml</td>
</tr>
<tr>
<td>5th</td>
<td>pl</td>
<td>do</td>
<td>jsp</td>
</tr>
</tbody>
</table>

The order of popularity using the unique file extension per page provides a different insight. In this case, “php” and “asp” are consistently the popular choice in all three snapshots, and “shtml” has dropped few places in all three snapshots. Through a comparison of the two tables, it can be concluded that web pages which allow dynamic scripts are commonly used in the web, and web pages with security measures such as “shtml” are linked extensively when they are being utilized in some webpages, but they are not widely used across many web pages. The trend of dynamic file extensions being widely used seems to be a permanent feature of the recent and even the current web.

### 3.5.3 Composition of the general Top Level Domains

An area of change not so well documented is in the Top Level Domains (TLD). A number of new TLDs were introduced over the years [131], Table 3.12 shows the TLDs introduced since 2000 where documentation was available. There are other new TLDs such as .tv .int .us. Besides the introduction of new TLDs, the composition of various TLDs has also changed through the years. The top 10 domains by host in 1997 were .com .edu .net .jp .uk .de .us .au .ca .mil, in decreasing order as represented in WT100G repository, and this order changed every year [138]. To investigate the change in TLD further, we compared the composition of various general TLDs...
in the 1997 Internet as represented by WT100G to those in the 2004 Internet, and the result is shown in Figure 3.6, where other are domains that do not contain any of the general TLDs. The general TLDs are ordered by composition in the 1997 Internet in the graph, and it is clear that the .com TLD is still about two-third of the total gTLD composition, whereas the composition of other gTLDs has mostly decreased after 7 years, to give rise to other TLDs.

Table 3.12: TLDs introduced after 2000

<table>
<thead>
<tr>
<th>Year</th>
<th>TLDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>.info .museum</td>
</tr>
<tr>
<td>2002</td>
<td>.name .coop</td>
</tr>
<tr>
<td>2003</td>
<td>.biz</td>
</tr>
<tr>
<td>2004</td>
<td>.aero .pro</td>
</tr>
</tbody>
</table>

The study conducted by [29] observed the change within various TLDs, and pointed out that less than 10% of web pages in most TLDs change everyday. However, more detailed assessments conducted as part of the research project revealed that the .com TLD is most frequently updated with 40% of pages changing everyday. In contrast, the most static TLDs are identified to be .edu and .gov, where 50% of pages did not change at all in a four month duration.

![Figure 3.6: The composition of various general TLDs in 1997 and 2004 respectively](image)

Although the various rates of change in different TLDs provide a general overview of the update frequency in the general TLDs, but the TLD may not be a fair method of categorization. It has been shown that .com is the most commonly used TLD. There are companies that dedicate themselves to provide news, to allow individuals to create personal web sites or to sell some products online, and they would all share the .com TLD. Therefore, in order to obtain more detailed information about the rate of change in the web, a different categorization was needed.
In the next analysis on the rate of change in the web, domains are grouped into seven categories according to their aim and purpose. The categories are commercial, education, government based, knowledge provision, news, personal and technical pages. Each category was examined in detail to observe the different properties in the pages and the type of change in these categories over a period of 4 months. The properties in the categories are summarized in Table 3.13.

Table 3.13: Properties of different types of domains

<table>
<thead>
<tr>
<th>Category</th>
<th>Size of domain</th>
<th>Depth of directory structure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
<td>max</td>
</tr>
<tr>
<td>Commercial</td>
<td>1</td>
<td>64</td>
</tr>
<tr>
<td>Education</td>
<td>34</td>
<td>3292</td>
</tr>
<tr>
<td>Government</td>
<td>1</td>
<td>8426</td>
</tr>
<tr>
<td>Knowledge</td>
<td>1</td>
<td>768</td>
</tr>
<tr>
<td>News</td>
<td>4</td>
<td>5160</td>
</tr>
<tr>
<td>Personal</td>
<td>4</td>
<td>646</td>
</tr>
<tr>
<td>Technical</td>
<td>1</td>
<td>210</td>
</tr>
</tbody>
</table>

From the properties displayed in Table 3.13, we found that educational domains are mostly large, and the size of government domains vary greatly from a single page to one which consists of 8426 pages. The directory structure of pages in different categories differ as well. Government, education and news categories tend to have deeper directory structures, while pages in commercial category have the most shallow structure in average. Now that the basic properties of these categories are made aware, the observation on the amount of change that occurred in a 4 months duration can be discussed.

Table 3.14: Rate of change in different types of domains

<table>
<thead>
<tr>
<th>Category</th>
<th>Average change in the entire category</th>
<th>Average change in index pages</th>
<th>Average change in intermediate pages</th>
<th>Average change in leaf pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commercial</td>
<td>0.96</td>
<td>0.910</td>
<td>0.000</td>
<td>0.994</td>
</tr>
<tr>
<td>Education</td>
<td>0.97</td>
<td>0.606</td>
<td>0.724</td>
<td>0.977</td>
</tr>
<tr>
<td>Government</td>
<td>0.98</td>
<td>0.941</td>
<td>0.776</td>
<td>0.991</td>
</tr>
<tr>
<td>Knowledge</td>
<td>0.99</td>
<td>0.642</td>
<td>1.000</td>
<td>0.998</td>
</tr>
<tr>
<td>News</td>
<td>0.92</td>
<td>0.626</td>
<td>0.871</td>
<td>0.920</td>
</tr>
<tr>
<td>Personal</td>
<td>0.77</td>
<td>0.867</td>
<td>0.671</td>
<td>0.998</td>
</tr>
<tr>
<td>Technical</td>
<td>0.96</td>
<td>0.888</td>
<td>0.909</td>
<td>0.976</td>
</tr>
</tbody>
</table>
Table 3.14 shows the average change in the entire category, the index pages, the intermediate pages and the leaf pages; where the average refers to the average dot-product score between the contents of webpages at the beginning and end of the 4 months period. A score of 1.0 would indicate no change at all, and 0.0 would indicate a total change of topic in the web page. The content of Table 3.14 indicates that different portions of the web experience a varied rate of change, and the rate of change is influenced by the purpose of the information. Not surprisingly, the personal web pages appear to undergo more changes with an overall average dot-product score of 0.77, followed by domains that offer news articles at 0.92, whereas web pages for the purpose of knowledge provision have the least amount of overall changes with an extremely high average of 0.99. The position of a page within a domain also affects the amount of change the page is likely to undergo. For example, index pages are expected to be updated frequently so as to communicate up-to-date information about its specific topic to the audience. As Table 3.14 shows, index pages indeed have a lower average than leaf pages, which means that more changes occur within index pages than in leaf pages. However, intermediate pages which are neither index pages nor leaf pages also experience significant changes in domains of personal, educational and government web sites. It should be pointed out that a score of 0 is recorded for intermediate pages in the commercial domain, but that is not caused by a total change of content in the 4 months, but the lack of intermediate pages. As Table 3.13 shows, the minimum depth is 1 and the maximum depth is 2 for commercial domains, therefore, the commercial domains investigated simply do not have intermediate pages.

Although this subsection established that web pages experience changes at different rates due to their different purposes, the types of change is not yet known; therefore, a more detailed assessment in the changes within web pages will be provided in the next subsection.

### 3.5.4 The type of changes in the page content

After the investigation which provided awareness that webpages experience different rates of change, the amount and type of changes in the document content should also be analyzed. Roach [108] pointed out that the popularity of the Internet has encouraged a wide variety of subjects, and the trend is likely to continue. An empirical content analysis was conducted on the accessible pages from the WT10G collection, comparing the vocabulary of the pages from the collection to those from the current web. The dot product of the vocabulary vector from the old and new versions of the webpage on the same web address was taken, to produce a score from 0 to 1, where 0 indicates a total change of subject and 1 indicates no change.

As was described earlier in this chapter, and as was seen in Figure 3.1, there are more than
two-thirds of pages which have had a significant change in content (a dot product score of $< 0.3$), implying a change in the subject area of those web pages. It is possible that in some cases, the nature of a web page is to be frequently updated with a variety of content, such as those for news publishing purposes; therefore, a follow-up investigation was conducted, which examines the number of unique words in a web page. The investigation revealed that more than a half of the pages have had a dramatic growth or reduction in page content, where the difference in the number of unique words is more than 50. Approximately $59\%$ of those pages had an increase of unique words, and $34\%$ experienced a decrease. This suggests that a larger proportion of web pages are updated with a content that provides more information.

In addition, there were 1134 pages with a dot product score of 0.3 to 1, these web pages remained focused in a similar subject area. $10\%$ of these pages had a growth of more than 50 unique words, indicating a significant addition of information on existing content or the broadening of subject area in the web page content. The phenomenon of increasing variety of subjects observed by [108] in 2004, has been confirmed through an empirical analysis conducted in 2006 as part of the research project. This trend is expected to continue into the future web. An additional observation made during the analysis revealed the trend of increasing document size to accommodate more information on web pages. This is also a probable trend.

### 3.6 Discussion and conclusion

In this chapter, some of the existing testbeds available for information retrieval experiments were examined for their suitability as the testbed for experiments of this research project. It was decided that it would be more appropriate to construct new testbeds as it has a number of benefits. Firstly, the behaviour and performance of the crawler can be monitored and assessed through the data retrieval task from the web, and the retrieved data from such a task can be processed to form testbeds, instead of being discarded. Secondly, the construction of new testbeds also provides the awareness on the amount of storage required for storing the same number of webpages as older testbeds, as well as the amount of processing power and time required. Thirdly, the construction of new testbeds also allowed for a comparison with older testbeds, and this reveals the amount and types of changes that occur in the web over a period of time. Lastly, the newly constructed testbeds can be stored and used for future experiments.

A centralized distributed crawler was utilized to retrieve snapshots of the web to form the basis of testbeds. The newly developed crawler allowed web data to be obtained in an efficient and complete manner, which provided a perspective into the World Wide Web, which is different
from what is possible through other crawlers. As a result, the testbeds were successfully con-
structed, with the added advantage of allowing identification of trends through analysis, which
allowed the opportunity to estimate the characteristics of the future web.

As mentioned previously, the properties of the web can be better observed through the con-
struction of new testbeds, which would assist in designing a quality-focused information retrieval
system well-suited for the web environment. The trends of a decreasing inaccessibility rate, the
high variation in file extension popularity, the relatively consistent composition of the general-
TLD, and a gradual addition of web page content were identified. From this, it was decided that
the crawler would not crawl exhaustively as a significant portion of the web cannot be retrieved
from the dynamic web which has such a high growth rate. Instead, the aim should be to contain a
large portion of target data in the retrieved collection. The target data, in the case of this research
project, is data which meets a quality standard.

The material in this chapter on the analysis of the WT10G benchmark testbed has been pub-
lished in the 14th International World Wide Web Conference in 2005 [28], and the material on
the crawler implementation used as the instrument of data collection has been published in the
International Conference on Web Intelligence in 2008 [71]. The focus in this chapter and in the
thesis so far has mostly been on the information retrieval component of the research; the quality
component of the research has not yet been discussed. The next chapter will begin to provide
some initial analysis and discussion on how the quality aspect of the research can be addressed.
The chapter will describe the investigation on how web documents can be grouped, and whether
it is possible to form groupings based on their level of quality.
Chapter 4

Document grouping through clustering

4.1 Introduction

The previous chapter showed a number of testbeds which could be used for the experiments in this research, as well as the approaches taken in the collection of a web snapshot as a more up-to-date testbed. This chapter will look at ways in which meaningful groupings of documents in the testbed can be carried out. The investigation into possible grouping techniques is deemed necessary, as before the design and implementation of a quality evaluation approach, some preliminary investigations are required. These preliminary investigations aim to understand the document characteristics and relationships among documents on the Web.

One approach which was taken in this research project for the identification of document characteristics and relationships is to perform document grouping through clustering. The unsupervised machine learning task of clustering allows documents with similar properties to be located on a map in close proximity to each other, and therefore form a cluster without external influence.

It has already been shown in Section 2.2 that the Web can be grouped in different levels of granularity, according to the domain, site or the TLD. However, these groupings only focus on the location of the web documents. This chapter aims to investigate whether a different type of grouping can be formed, and whether those groupings can provide some indication about the characteristic similarity or even the quality of the documents. To assist this new type of grouping based on characteristics of documents, the internal structure will be used. The internal structure of a document will indicate the organization of contents and occurrences of document elements without analyzing the semantics of the document. This corresponds to the understanding of quality as independent to the topic area of documents as defined in Section 2.7.

As mentioned in the previous chapter, a testbed is required for conducting experiments de-
signed for the Web due to the size and dynamic nature of the Web. One of the testbed that was available for the research project is the INEX challenge 2006 testbed in XML format. The INEX 2006 XML testbed was considered potentially useful for this research project due to the similarity between the structure of XML and web-based HTML documents, the testbed also has target information, which allows experimental results to be evaluated. Therefore the INEX 2006 XML testbed is used as a testbed for the preliminary experiments described in this chapter. This chapter is organized by firstly describing the proposed approaches, then detailing the experimental settings and results. A discussion and comparison will then follow, and finally the chapter will conclude with a brief summary.

4.2 Proposed approaches

In general, structured objects such as XML or HTML documents can be described by graphs, e.g. acyclic directed graphs, cyclic graphs, un-directed graphs, etc. Graphs are generalizations of the more common vectorial representation as a graph can encode relationships among structural elements of objects, or provide contextual information concerning data points which may be described in vectorial form.

It is recognized in the research community that any model which is capable of dealing with structured information can potentially be more powerful than approaches which are limited to the processing of vectorial information. This observation motivated the development of machine learning methods which are capable of encoding structured information. A noteworthy result of such an effort is the Graph Neural Network (GNN), which is a supervised machine learning method capable of learning from a set of graphs [110]. The GNN is one of the more powerful supervised machine learning methods devised since it is capable of processing arbitrary types of graphs, e.g. cyclic, un-directed, where (numeric) labels may be attached to nodes and links in the graph. In other words, a GNN can encode the topology of a given set of graph structures as well as the numerical information which may be attached to the nodes or links in the graph.

Supervised machine learning methods require the availability of target information for some of the data, and are typically applied to tasks requiring the categorization or approximation of information. Unsupervised machine learning methods have no such requirement on the target information, and are typically applied to tasks requiring the clustering or segmentation of information. Unsupervised machine learning techniques for graph structured information are often based on the well-known Self-Organizing Maps [78] and are called Self-Organizing Maps for Structured Data (SOM-SD) [52]. While a SOM-SD is restricted to the processing of bounded
positional acyclic directed graphs, it is found that this is sufficient for many practical applications. The introduction of a contextual SOM-SD (CSOM-SD) extended the capabilities of the SOM-SD model to allow for the contextual processing of bounded positional directed graphs which may contain cycles [54]. SOM, SOM-SD and CSOM-SD are described in Sections 4.2.1, 4.2.2 and 4.2.3 respectively.

At the INEX 2005 workshop it was shown for the first time that a SOM-SD based model can produce exceptional performances on XML classification tasks by winning the 2005 international competition on XML classification task. The experiments discussed in this chapter applies the SOM-SD and CSOM-SD to a new and relatively large XML clustering task in 2006 to investigate the suitability of such machine learning methods - a task that has never before been executed in this manner. This chapter will also answer the question of whether the contextual processing of information can lead to improvements in this XML mining task. This question can be answered by comparing the SOM-SD which processes information in a strict causal manner, with results from applying the CSOM-SD which processes the same data in a contextual fashion.

4.2.1 Self-Organizing Maps

This section gives an overview to how unsupervised learning can be achieved when using Self-Organizing Map techniques. Unsupervised neural methods are trained on data for which target information is not available. The general application of such methods is to cluster data such that data that share certain properties are clustered together; a method which is also known as *self-organization* method. The most famous and most widely used of the unsupervised learning methods are Self-Organizing Maps (SOM) originally developed by T. Kohonen in 1989 [78], and its variants.
The basic idea of SOM is simple. The SOM defines a mapping from high $n$-dimensional input data space onto a regular lower $q$-dimensional array (often two-dimensional array) called a display map. The intersection of the two dimensional grid in the display map is represented by an entity called a neuron. Every neuron $i$ of the display map is associated with an $n$-dimensional codebook vector $m_i^T = (m_{i1}, \ldots, m_{in})^T$, where the superscript $T$ denotes the transpose of a vector, and $m_{ij} \in \mathcal{R}$. The neurons on the map are connected to adjacent neurons by a neighborhood relation, which defines the topology, or the structure, of the map. The most common topologies in use are rectangular and hexagonal [78]. Adjacent neurons belong to the neighborhood $N_i$ of the neuron $i$. Neurons belonging to $N_i$ are updated according to a neighborhood function $f(\cdot)$ such as a Gaussian-bell or a Mexican-hat function [78]. Typically, the topology and the number of neurons remain fixed from the beginning of the training process. The number of neurons determines the granularity of the mapping, which has an effect on the accuracy and generalization capability of the SOM [78]. Figure 4.1 shows the architecture of a two-dimensional SOM of size $8 \times 4 = 32$. Each hexagon represents a neuron location, and since each neuron has six neighbors, the topology of this map is hexagonal. Also shown in Figure 4.1 is that each of these neurons is associated with an $n$-dimensional codebook vector $m_i$.

The vector $m_i$ are updated by a training process. During the training phase, the SOM forms an elastic cover that is shaped by input data. The training algorithm controls the cover so that it strives to approximate the density of the underlying data. The reference vectors in the codebook drift to areas where the density of the input data is high. Eventually, only few codebook vectors lie in areas where the input data is sparse. The result is that the input data is clustered. The training algorithm for the weights associated with each neuron in the $q$-dimensional lattice ($q = 2$ in our case) can be trained using a two step process as follows:

**Competitive step:** One training sample $u \in \mathcal{R}^n$ is randomly drawn from the input data set and its similarity to the codebook vectors is computed:

$$ r = \arg\min_i \|u - m_i\| $$

where $\|\cdot\|$ is the Euclidean distance.

**Cooperative step:** After the best matching codebook $m_r$ has been found, all codebook vectors are updated. The winning vector $m_r$ itself as well as its topological neighbours are moved closer to the input vector $u$ in the input space. The magnitude of this adjustment is governed by a learning rate $\alpha$ and by a neighborhood function $f(\Delta_{ir})$, where $\Delta_{ir}$ is the topological distance between $m_r$ and $m_i$. As the learning proceeds and new input vectors are given to the map, the learning rate gradually decreases to zero according to a specified
function. Along with the learning rate, the neighborhood radius decreases as well \(^1\). The updating algorithm is given by:

$$\Delta m_i = \alpha(t)f(\Delta_{ir})(m_i - u)$$

(4.2)

where \(\alpha\) is a learning coefficient, \(f(\cdot)\) is a neighborhood function which controls the amount the weights of the neighboring neurons is updated. The neighborhood function \(f(\cdot)\) can take the form of a Gaussian function:

$$f(\Delta_{ir}) = \exp\left(-\frac{||l_i - l_r||^2}{2\sigma(t)^2}\right)$$

(4.3)

where \(\sigma\) is the neighborhood radius, and \(l_r\) is the location of the winning neuron, and \(l_i\) is the location of the \(i\)-the neuron in the lattice. Other neighborhood functions are possible.

The two steps together constitute a single training step and they are repeated until the training process ends. There are a number of ways in which the training can terminate. One way is to fix the number of iteration steps. In this case, the number of training steps must be fixed prior to training the SOM because the rate of convergence in the neighborhood function and the learning rate are calculated based on this information.

The SOM, and in fact most conventional ML methods, require data to be available in vectorial form. This includes recurrent neural network methods which can process continuous signals by processing shifting fixed sized sub-sections of a continuous signal one at the time. A recent extension [52] produced Self-Organizing Maps which can process graph structured data without requiring the pre-processing of such data. This will be addressed in the following subsection.

### 4.2.2 Self-Organizing Maps for Structured Data

Approaches to enable SOMs to map graph structured information were proposed relatively recently in [52, 53]. Work on SOM methods capable of mapping structures was inspired by developments in supervised neural networks for graphs [46]. The basic idea behind these approaches is to process individual nodes of a graph rather than the graph as a whole, and to provide a mechanism to incorporate structural dependencies between the nodes. This is achieved by adding a set of \(states\) which stores information about the activations of the network in processing a given node, and to use the states associated with neighboring nodes as an additional inputs in processing the current node. The recursive application of this approach ensures that information about the encoding of any node in a graph is passed on to any other node in the graph as long as there

---

\(^1\)Generally, the neighborhood radius in SOMs never decreases to zero. Otherwise, if the neighborhood size becomes zero, the algorithm has no longer topological ordering properties ([78], page 111).
is a path connecting these nodes. In other words, the idea allows to encode a graph as a whole by processing individual nodes one at a time.

When applied to SOM, the idea allows the processing of labeled directed acyclic graphs (labelled tree structures) [52] by producing the network input $x_v$ for every node $v$ in a graph through the concatenation of the label $u_v$ and states $y_{ch[v]}$ so that $x_v = [u_v, y_{ch[v]}]$. The $y_{ch[v]}$ are simply the mappings (the coordinates) of the children of $v$.

Training a SOM-SD is performed in a similar manner to the classical approach [78]. The difference is that for computing the similarity in Equation (4.1), the measure needs to be weighed so as to account for the hybrid data in the vector $x_v$. Also, some components (the states) of the input vector need to be updated at every training step. The resulting training algorithm is as follows:

**Step 1** A node $v$ is chosen from the data set. When choosing a node special care has to be taken that the children of that node have already been processed. Hence, at the beginning the terminal (sink) nodes of a graph are processed first, the root (source) node is considered last. Then, vector $x_v$ is presented to the network. The winning neuron $r$ is obtained by finding the most similar codebook entry $m_r$ as follows:

$$r = \arg \min_i \| (x_v - m_i) \Lambda \|$$  

(4.4)

where $\Lambda$ is a $n \times n$ dimensional diagonal matrix. Its diagonal elements $\lambda_{11} \cdots \lambda_{pp}$ are set to $\mu_1 \in [0; 1]$, all remaining diagonal elements are set to $(1 - \mu_1)$. The constant $\mu_1$ controls the contribution of the data label component and the state vector component to the Euclidean distance.

**Step 2** After the best matching neuron has been found, the winning codebook vector and its neighbours are updated as in Equation (4.2).

**Step 3** The coordinates of the winning neuron are passed on to the parent node which in turn updates its vector $y_{ch}$ accordingly. This step neglects the potential changes in the state of the descendants due to the weight change in step 2. This approximation does not appear to cause any problems in practice [52].

Cycles of Steps 1 to 3 are executed repeatedly until a given number of training iterations is performed, or when the mapping precision has reached a given threshold.

The optimal choice of the values $\mu_1$ depends on the dimension of the data label $l$, the magnitude of its elements, the dimension of the coordinate vector $c$, and the magnitude of its elements.
The Euclidean distance in Equation (4.4) is computed as follows:

\[ d = \mu_1 \sum_{i=1}^{p} (u_i - m_i)^2 + (1 - \mu_1) \sum_{j=1}^{2o} (y_{ch,j} - m_{n+j})^2 \]  
(4.5)

where \( o \) is the number of children whose states are fed into the node \( v \). This value changes with each node according to the connectivity of the node \( v \). The factor 2 is due to the fact that each child node will have two coordinates. Hence, it becomes clear that the sole purpose of \( \mu_1 \) is to balance the influence of the two terms in the behaviour of the learning algorithm. Ideally, the influence of the data label and the coordinate vector on the final result is equal. This can be computed by statistically analysing the training dataset [52].

It can be observed that the SOM-SD processes data in a strict causal manner from the terminal nodes towards the root node. Processing in the reverse direction, i.e. from the root node towards the terminal nodes is also possible but rarely applied. It is not possible to process nodes in both directions or in a random order since otherwise in Step 2 it is possible that not all states of all of a particular selected node’s neighbors are available. An extension which circumvents this problem is described in the following section.

4.2.3 The contextual Self Organizing Maps

The SOM-SD processes nodes in an inverse topological order (i.e. from the terminal nodes towards the root node) which is required to guarantee that the states of all dependencies are available at any time during network training. This limits the ability of the SOM-SD to discriminate certain sub-structures. For example, the mapping of a graph with a single node \( A \) would be exactly the same for a node \( A \) that occurs as a leaf in another graph with many nodes. This is because no information about parent nodes can be included when computing a mapping and hence, any identical sub-structure would be mapped onto the same location independent of the contextual arrangement in which such a sub-structure occurs.

In order to capture differing contextual arrangement of nodes in a graph it is necessary to allow the inclusion of information about the mappings of its parent nodes as well as the mapping of child nodes at any time during training. A solution to this problem would bring a far reaching advantage: it would become possible to process undirected or cyclic graphs, a much more general class of graphs than tree structures.

A first solution was proposed in [53]. In [53] it is observed that while the mapping of parent nodes is not available at a time instant \( t \) that it is available by retrieving the mappings at time \( t - 1 \). Given the asymptotic nature of the training algorithm it can be assumed that mappings
do not change significantly between two iterations\(^2\), and hence, the utilization of mappings from a previous iteration in order to fill the states that are not available at a current training time (iteration) should be a valid approximation.

An advancement of this idea removed two obstacles: (1) the initialization problem (i.e. at time \(t = 0\) there is no mapping from time \(t - 1\) available), and (2) the need to memorize the mappings of the time instant \(t - 1\) by recursively computing a stable point in the state space [54]. The method proposed in [54] allows to process nodes in a random order thus removing the necessity of having an acyclic directed graph, and this also removes the need to sort the nodes in a particular order before processing can commence. The proposed mechanism to compute the states of all parents and children of a randomly selected node is by computing a stable point in the mapping of nodes as follows:

A Select a node from the dataset. Generate a \(k\) dimensional vector. Initialize the first \(p\) elements with the data label that is attached to this node. Initialize all the states of offsprings \(y_{ch[v]}\), and all the states of parents \(y_{pa[v]}\) with zero since these states are as yet unknown. Initialize all remaining elements with \((-1, -1)\) (corresponding to missing parents or missing children.)

B Find the best matching codebook entry.

Steps A and B are executed for each node in the training set. This computes every node’s state as far as it is possible at this stage. A stable point is computed by recursively executing the following steps: (a) select a node from the dataset; (b) generate a \(k\)-dimensional vector with the first \(p\) elements initialized with the data label that is attached to this node, initialize \(y_{ch[v]}\) with the states of this node’s offsprings (which is available from the previous iteration), the \(y_{pa[v]}\) with the states of this node’s parents (which are available from the previous iteration), and all other elements with \((-1, -1)\) as before. Once every node in the training set has been considered, this is called “one iteration”. The algorithm iterates until the number of changes in the node states during an iteration do not decrease further. With this mechanism in place, it becomes possible to train a CSOM-SD as follows:

Step 1 Compute the stable point.

Step 2 Train the SOM-SD on every node in the training set as usual ensuring that the vector components \(y_{ch[v]}\) and \(y_{pa[v]}\) are updated by using the states from the previous iteration,

\(^2\)This is particularly the case during the final stages of the learning procedure.
and ensuring that the components $I_v$, $y_{ch[v]}$, and $y_{pa[v]}$ are weighted so as to balance their influence on the Euclidean distance measure similar to that described in Section 4.2.2.

Step 2 is iterated at least $N_d$ times, where $N_d$ is the number of training iterations.

Note that this approach, given a sufficiently large map, ensures that identical substructures which are part of different graph structures are mapped to different locations. It can be assumed that a map properly trained will map vertices to the same or nearby location only if the underlying graph structure is similar. Note also that the complexity of the training algorithm increases linearly with the size of the training set, the size of the network, and the maximum number of iterations. Hence, the approach provides a mechanism which may be capable of finding similar graphs (inexact graph matching) in linear time.

Note also that the stable fixed point is computed only once. One may consider the possibility of computing the stable fixed point every time the network parameters are adjusted (i.e. after processing a node). However, this would increase the computational complexity to a quadratic one, and may lead to a moving target problem.

It was shown in [54] that the CSOM-SD algorithm is stable in practical applications and it was found in a number of experiments that the CSOM-SD is consistently less sensitive to initial network conditions and training parameters in comparison to SOM-SD. There is as yet no explanation of such behaviours.

### 4.3 Experiments and results

The experimental results for the clustering tasks are compared using precision and recall as in Micro F1 and Macro F1 as a measure. The F1 measure is based on the common information retrieval performance measures of precision and recall, but F1 is a single measure that takes both the precision and recall into account. Therefore is a more complete view of the machine learning performance.

In a machine learning task that attempts to categorize data into classes, we have a set of positive data (positives) which refers to the set of documents that was assigned to a particular category by the learning process, and the other documents are considered negative data (negatives) for that particular category. Then the assignment is compared to the the correct category that the documents belong to. The results is the following four types of documents for each category:

- True Positive (c) - The positives assigned to the particular category correctly
• False Positive (d) - The positives that were assigned to the particular category incorrectly

• True Negative (e) - The negatives that correctly do not belong to the particular category

• False Negative (f) - The negatives that were not, but should have been assigned to the particular category

From these groupings, Precision and recall can be defined as: Precision \( p = \frac{c}{c+f} \), Recall \( r = \frac{c}{c+e} \). It can be seen that precision and recall complement each other in the sense that manipulating the result to increase one could result in a poor performance in the other. For example, assigning all documents to all categories may produce a perfect recall, but will result in poor precision; similarly, limiting the number of assignments could result in a high precision, but the recall performance will be sacrificed. This is the reason that F1, a combination of both precision and recall is used. F1 measure is defined as: \( F1 = \frac{2 \times p \times r}{p+r} \).

Since the learning problem involves 18 unbalanced classes, a fair comparison measure should be adopted so that the performance of a smaller class could have a similar effect as the performance of a larger class, instead of being somewhat neglected. Such a balanced measure is achieved through Micro F1 and Macro F1, where the average over all the classes is taken. Micro F1 is weighted by the number of documents in each class, so may be strongly influenced by large classes in an unbalanced testbed. However, Macro F1 is non-weighted, so the F1 measure for each class is calculated individually, then summed up and divided by the total number of classes.

It can be seen that F1 is a more balanced performance measure than precision or recall. Therefore the experimental results in this chapter will display the micro and macro F1 performances to allow an insight into the weighted and unweighted clustering results.

4.3.1 Analysis of the INEX XML testbed

The experiments are performed on the INEX testbed which includes XML formatted documents, each from one of the 18 different classes which correspond to actual journals, covering both transactional and non-transactional journals and across a number of topics as can be seen in Table 4.1. However, there are up to 5 journals that belong to the same structural (transactional or non-transactional) and semantic (topics) grouping, therefore distinct differences cannot be expected from documents of several journals. Furthermore, the journals are unbalanced in the number of documents they contain, therefore, this learning task is high in complexity, yet contains features that are commonly found in real-world problems.

The documents used in the training process is the training portion of the dataset, which consists of 6053 documents; and the number of XML tags in each document ranges widely, from...
Table 4.1: The topic area and types of journals in the INEX 2006 XML test-bed

<table>
<thead>
<tr>
<th>Groupings</th>
<th>Computer</th>
<th>Graphics</th>
<th>Hardware</th>
<th>Artificial</th>
<th>Internet</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactional</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Non-transactional</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

9 to 7024 pairs of tags, with a total of 3,966,123 pairs of tags. To represent the structure of a document, a tree could be used where each node in the tree represents the occurrence and the location of the XML tags. This will result in large trees where the maximum depth is 19 and the maximum out-degree is 1023, where the out-degree refers to the number of child nodes directly under a particular node. Another observation of the INEX test-bed is that there is a total of 165 unique tags, and some tags occur with a high frequency in a number of documents, but not all tags occur in all documents.

The documents used in the testing process is the testing portion of the testbed, which consists of 6054 documents. The documents in the testing set has no overlap with the documents in the training set, but the proportion of documents in each journal class is comparable to the training data, to ensure that the rules learnt from training can be applied to the test data and that a similar level of performance can be expected.

The clustering task for the INEX testbed consists of two components: clustering using only the structural features, and clustering using both the structural and textual content. The two tasks vary in the degree of difficulty. The following section describes the proposed approaches for this clustering problem.

The motivation for considering clustering in this thesis is as follows:

- Recently developed machine learning approaches are capable to encode graph structured information. Moreover, the machine learning methods considered here scale linearly with the number of documents (i.e. Web pages), and hence, are a scalable approach to the processing of Web documents. The Web is naturally represented as a graph with hyperlinks defining a directed link between two nodes which represent the associated Web pages.

- We consider the possibility of clustering Web documents into various clusters where each cluster should represent documents of a certain quality. While the INEX competition does not aim at clustering documents by quality, this exercise will provide us with some results that indicate the suitability of the approach towards clustering of generic Web documents.
into clusters of various quality.

4.3.2 Data Pre-processing

While not strictly necessary, we opted to pre-process the dataset so as to reduce the turn around time for the experiments. More specifically, we reduced redundancies in the dataset so as to speed up training times.

As described in the data analysis, the training data has a total of almost 4 million nodes and a maximum out-degree of 1,023. The XML documents are represented by trees with the nodes denoting XML tags and the arc between a parent and a child representing the encapsulation of a tag in another tag. Given a learning method which processes each node individually, and requires the inclusion of the states of all offsprings, it is found that such data could result in very long training durations.

In particular, the input dimension grows twice as fast as the (maximum) number of children at any node in a graph. The dimension of codebook vectors grow at the same rate. Since these vectors are multiplied as in Eq.4.5, the computational complexity grows quadratic with the number of children. Note that the computational complexity grows linear with the number of nodes in the graphs, and hence, any optimization in the representation of graphs should address the outdegree.

The training process using the unprocessed structure is estimated to take 6818 days, which is not desirable. Therefore, while in general the learning method does not necessarily require pre-processing steps, some pre-processing is applied in order to improve the turn around time for the experiments. Four pre-processing approaches were considered and are compared.

1. Consolidating repetitive sub-structures: This was suggested by an approach taken at the INEX-2005 challenge [56], which consolidates repetitive sub-structures. However, due to the large variation of structures and tag positions in the set of INEX-2006 documents, this approach did not significantly decrease the expected training times. The estimation for training duration after this pre-processing is 1698 days, which is still unreasonably long. Another drawback of this approach is that repeating sub-structures may be a significant feature, and compressing that feature could impact the clustering performance.

2. Constructing tag-based connectivity graph: Construct a connectivity graph based on the unique tags in each document instead of using a tree to represent the document structure. In the connectivity graph, each unique tag is represented by a node, so multiple occurrences of the same tags are merged. Also, care was taken to unfold the cyclic portions of the
connectivity graph using the depth-first links visited method. The training time estimation for this approach is 70 hours, which is still too long.

3. Extracting header segment of documents: Segment documents by using only the structure of a chosen segment. The document is first segmented by the nodes in the first level of the structure tree. This identified the substructures: FNO, DOI, FM, BDY and BM, where FNO and DOI do not contain any XML tags, and BM does not occur in all documents, so that leaves us with FM and BDY. The substructure where the maximum out-degree occurred was in the BDY segment, so we decided to take the FM segment, which is small enough to train without any processing of its structure. The estimated training time is 16 hours, which is reasonable for training, so this structure is used for the structure only clustering task.

4. Developing a basic framework: This approach considered the use of a simplistic framework. While the training duration using the structure of document headers may be reasonable for structure only clustering, it will be far too long for clustering based on structure and textual information, where the challenge of effectively encoding high dimensional textual information into the structure needs to be addressed. As a result, a framework is developed where only the key tags of documents are included to minimize and control the number of nodes per document. The key tags refer to significant structural elements of each document such as Title, Abstract, Body, and Conclusions. Thus, the result is a reduction of the structural representation of each document to at most four nodes. This is explained in some more detail in Section 4.3.4. This approach can be trained in as little as 8 minutes before adding the textual information and approximately 9 hours with textual information.

4.3.3 Training using Structural Information

For the clustering using only structural information, the documents went through the header structure extraction process (the third approach in Section 4.3.2). This approach has the least structural modification, therefore is expected to closely resemble the original document structure.

A common problem with Self-Organizing Maps is that training parameters need to be determined using a trial and error approach. This paper proposes a more sophisticated approach by statistically analysing the properties of the dataset which helps to make a reasonable assumption about the underlying difficulty of the learning problem. Then training parameters which should be most suitable for this given task are set. Absolutely no information about target values or class memberships are used during this task, and hence, the approach remains unsupervised.
The SOM provides a discrete mapping space. The size of the map needs to be determined prior to a training session. In order to obtain an indication of suitable network sizes, it is recommended to consider the analysis of the training set. It is found that the set which will be used for the experiments consists of 108,523 nodes. Each node is the root of a sub-structure. Thus, there are 108,523 sub-structures in the dataset, 2,275 of these sub-structures are unique, while 48,620 of the nodes are found in a unique contextual arrangement within the graphs. Since, each pattern is represented by a codebook, the number of codebooks should be large enough to represent every different feature of any pattern. In fact, usually the number of codebooks is selected approximately equal to the number of different sub-structures in SOMSD and unique graph-node pairs in CSOM-SD. In other words, the statistical analysis of the dataset suggests that a SOMSD would require at least 2,275 codebook vectors in order to provide the means to differentiate the mappings of the unique features in the input dataset. It furthermore tells us that a CSOM-SD would require at least 48,620 codebooks. Hence, it is observed that the CSOM-SD should be able to differentiate the mappings much more effectively than when compared to the SOM-SD.

For first experiments we utilized maps which consisted of 8,800 codebook vectors. Training was conducted using both SOM-SD and CSOM-SD. The training parameters used are as shown in Table 4.2.

<table>
<thead>
<tr>
<th>Clustering method</th>
<th>Map size</th>
<th>Learning rate</th>
<th>Iteration</th>
<th>Radius</th>
<th>$\mu_1$</th>
<th>$\mu_2$</th>
<th>$\mu_3$</th>
<th>Training time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM-SD</td>
<td>8800</td>
<td>0.7</td>
<td>150</td>
<td>15</td>
<td>0.05</td>
<td>0.95</td>
<td>0</td>
<td>16 hours</td>
</tr>
<tr>
<td>CSOM-SD</td>
<td>8800</td>
<td>0.7</td>
<td>100</td>
<td>15</td>
<td>0.005</td>
<td>0.095</td>
<td>0.9</td>
<td>16 hours</td>
</tr>
</tbody>
</table>

The initial training used a map size based on the number of unique substructures in the document structure tree. However, the map size that provides a good performance within allowable time is slightly larger than their initial map size which in this case is 8,800.

The use of appropriate parameters for training is essential for achieving good clustering performance. Usually, the best training parameters are identified through a trial-and-error process. It was observed that an initial learning rate of 0.7 and a large number of iterations is best for most training runs. However, the balance between high number of iterations and long training duration is a challenging task.

Another observation is that although previous clustering experience indicated that best performance can be obtained when the radius is a half of the length of the training map’s side, which
was not the case for this clustering task. For the structures used in clustering the INEX dataset, the radius that delivers best clustering performance remains small regardless of the map dimension. This could be due to the fact that nearby clusters are somewhat independent of each other and should be handled differently; therefore updating an extended radius causes the performance to decline.

The most influential parameter is perhaps the use of an appropriate weight value for the node label ($\mu_1$), as it also determines the weight of the children nodes ($\mu_2$) in the structure. The selection of appropriate weight for the node label is dependent on the importance of the node label in the structure used.

After the training process, each of the documents of the 18 different classes are assigned to a coordinate on the map, which is determined by the activated neuron for the particular graph. The results of such a mapping can be visualized in a reduced, 2-dimensional map. The best result obtained from SOM-SD experiments can be seen in Figures 4.3.3, 4.3.3 and 4.3.3. Note that the mapping is separated into 3 different plots due to the large number of classes, which would otherwise not be able to be distinguished. The figures show that there is no formation of clear clusters for each of the document classes. This provides an indication on the difficulty of this clustering task.

![Figure 4.2: Visualization of the mapping for documents of classes 1-6](image)

The parameters used for SOM-SD were taken as the starting point for CSOM-SD training and the map size is approximately the same as the number of unique nodes. Then, adjustments were made based on the observations of SOM-SD and CSOM-SD in the work from the previous
Figure 4.3: Visualization of the mapping for documents of classes 7-12

Figure 4.4: Visualization of the mapping for documents of classes 13-18
challenge, which indicated the following:

- The number of training iterations for CSOM-SD does not need to be as high as that used for SOM-SD.
- The map size for CSOM-SD should be much larger than the map size provided for SOM-SD.

The difference between the number of unique substructures and unique nodes indicated that for this particular structure, a CSOM-SD experiment that will require a map size that is about 15 times larger than that used for the SOM-SD. However, due to the time constraint, the CSOM-SD experiments conducted used only a map size as large as 8,800.

Similar to SOM-SD, the appropriate weight values are important for good CSOM-SD clustering performance. The difference in performance with various weight values on parent nodes ($\mu_3$) is quite significant. In an example where all parameters remain constant and only the weights were adjusted, an increase of weight on parent node from 0.5 to 0.9 increased the micro F1 clustering result by 2%.

Performance measures will be given by F1. Macro and micro statistics are also computed. Macro averaging is calculated for each cluster and then averaged while micro averaging is computed over all neurons and then averaged. $F1$ is defined as $\frac{2PR}{P+R}$ where $P$ is the precision and $R$ is the recall. $F1$ can be computed if target information is available for the training and test dataset by computing $R$ and $P$ as follows:

**Recall:** Assuming that for each XML document $d_j$ the target information $y_j \in \{t_1, \ldots, t_q\}$ is given. Since each XML document is represented by a tree, and since both the SOM-SD and CSOM-SD consolidate the representation of a tree in the root node, we will focus our attention just on the root of the tree. With $r_j$ we will refer to the input vector for SOM-SD or CSOM-SD representing the root of the XML document $d_j$. Then the index is computed as follows: the mapping of every node in the dataset is computed; then for every neuron $i$ the set $win(i)$ of root nodes for which it was a winner is computed. Let $win_t(i) = \{r_j | r_j \in win(i) \text{ and } y_j = t\}$, the value $R_t = \max |\frac{|win(i)|}{|win(i)|}\}$ is computed for neurons with $|win(i)| > 0$ and we obtain $R = \frac{1}{W} \sum_{i,|win(i)|>0} R_t$, where $W = \sum_{i,|win(i)|>0} 1$ is the total number of neurons which were activated at least once by a root node.

**Precision:** $P = \sum_{i,|win(i)|>0} \frac{R_t|win(i)|}{W}$. 

119
Table 4.3: Training parameters used for the clustering of both structural and textual information

<table>
<thead>
<tr>
<th>Clustering method</th>
<th>Map size</th>
<th>Learning rate</th>
<th>Iteration</th>
<th>Radius</th>
<th>$\mu_1$</th>
<th>$\mu_2$</th>
<th>$\mu_3$</th>
<th>Training time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM-SD using 1-D textual label</td>
<td>40000</td>
<td>0.7</td>
<td>120</td>
<td>20</td>
<td>0.002</td>
<td>0.95</td>
<td>–</td>
<td>1.5 hours</td>
</tr>
<tr>
<td>SOM-SD using 3-D textual label</td>
<td>40000</td>
<td>0.7</td>
<td>120</td>
<td>20</td>
<td>0.001</td>
<td>0.999</td>
<td>–</td>
<td>10 hours</td>
</tr>
<tr>
<td>CSOM-SD using 1-D textual label</td>
<td>90000</td>
<td>0.7</td>
<td>100</td>
<td>10</td>
<td>0.003</td>
<td>0.332</td>
<td>0.665</td>
<td>20 hours</td>
</tr>
</tbody>
</table>

4.3.4 Training using Structural and Textual Information

For the task of clustering using structure and textual information, the input data is very different to the data for structure only clustering. The documents went through a simple framework filtering process (approach 4 in Section 4.3.2). This approach uses a maximum of 4 nodes to represent the document structure, so that a high dimension of textual information can be added without requiring excessive training time.

The clustering training was conducted using both SOM-SD and CSOM-SD. The training parameters used are as shown in Table 4.3.

As Table 4.3 shows, the simplicity of the framework allows various textual information to be added to the structure. The 1-dimensional textual label simply uses the number of unique keywords in the various segments of the document, whereas the 3-dimensional textual label contains information about the 3 keywords with the highest frequency in each document segment. Here, the keywords are dictionary words, where the dictionary lists all words found in the training set but has common words such as “the”, “is”, “a”, “in”, etc. removed.

The initial SOM-SD training used a map size based on the number of unique sub-structures in the document structure tree which was quite small. However, since the training time is not long, the map size was increased to spread out overlapping nodes on the map and obtain better clustering results.

Similar to structure only clustering, the provision of appropriate training parameters is important, so the best combination of learning rate, number of training iterations and map updating radius have been attempted for each input data.

Again, the most influential parameter is the use of an appropriate weight value for the node label ($\mu_1$), as it also determines the weight of children nodes ($\mu_2$) in the structure. The selection of appropriate weight for the node label is dependent on the importance of the node label in the
structure used. For this task, the existence and types of children nodes in this simple framework is quite important, therefore the weight on the node label ($\mu_1$) should remain low.

The difference between the unique nodes and the unique sub-structures for this framework is approximately 4 times larger, so the map size for CSOM-SD is expected to be 4 times larger than its SOM-SD counterpart. Although the performance obtained through the CSOM-SD is expected to be better than SOM-SD, however, the long training time and the resulting accuracy is a trade-off. Figure 4.3.4 shows the performances achieved by SOM-SD and CSOM-SD.

### 4.4 Discussion and comparison

The SOM-SD and CSOM-SD were applied to document mining tasks at INEX 2006. Both approaches produced winning results albeit amongst a fairly small group of participants [72]. However, it was observed that the CSOM-SD has a nonlinear computational complexity; in most cases, this is close to quadratic. This would limit the application of the CSOM-SD technique to small testbeds.

The test results for structure only clustering are given in Table 4.4, ordered by micro F1 in descending order. The test results for structure and textual clustering are shown in Table ???. The performance evaluation algorithm of Micro F1 and Macro F1 are as follows:

![Performance comparison between SOM-SD and CSOM-SD](image_url)

**Figure 4.5:** Performance comparison between SOM-SD and CSOM-SD when utilizing both structure and content information
The approaches taken by the other teams were as follows: (1) Team Doucet and Lehtonen from IRISA-INRIA, France, and from the University of Helsinki, respectively use K-means with different features (see Table 4.4 and Table ??). They adopt a two step procedure, first they cluster according to a set of features, then according to another. The feature $TE$ is “a structural indicator of the proportion of mixed content in an XML fragment”. (2) Team Tran, Nayak, Raymond from the Queensland University of Technology, Australia use a method called Progressively Clustering XML by Structural Similarity (PCXSS). The approach is an iterative method which uses a measure of the distance between a pair of trees which roughly counts the number of common paths from the root to the leaves. A cluster center is represented by a set of common paths. Clusters are built while reading the input trees. At each step of this procedure the input
tree is either inserted into the closest cluster or into a new cluster, if the minimum distance to the existing clusters is over a given threshold.

As the tables show, our team obtained the best performance in the structure only clustering. In fact, the performances obtained by using XML structure only also out-performed all the other teams involved in the structure and content clustering task (see Table ??). Although the CSOM-SD method does not appear to perform as well as the SOM-SD method, but bearing in mind that the map size provided for CSOM-SD is the same as the size used for SOM-SD, the CSOM-SD result may easily be improved.

Our performance in the structure and content could not achieve the similar standard, and we attribute this poor performance to the use of a significantly simplified structure which allowed the addition of textual information. However, the addition of textual information did not seem to add too much value to the clustering performance. This is perhaps due to the dramatic reduction of dimensionality for the textual information, which was carried out in order to keep the turn around time of the experiments reasonable.

SOMs have traditionally been useful for many data mining tasks due to the linear nature of the learning algorithm, their generalization ability, and due to their insensitivity to noise. With SOM-SD, this property is maintained with the exception of a quadratic dependence on the connectivity of nodes in a graph. The computational complexity grows quadratic with the outdegree a node in the dataset. In this paper it was shown that through the removal of redundancies in the data set it is possible to reduce the outdegree significantly, and hence, rendering SOM-SD practical for the given task. Thus, it was shown that the SOM-SD can be a suitable tool for the clustering of relatively large sets of documents. While in general it is not necessary to pre-process the data when using this method, we found that pre-processing can help to substantially reduce training times. It is noted, however, that the network, once trained, can respond very quickly to large sets of test patterns.

We have furthermore demonstrated that XML structure is causal. This means that a contextual processing of such data is unlikely to bring any advantages, and thus, machine learning methods can be optimized by learning the structure in one direction only.

The inclusion of textual information into the clustering task were addressed. The results obtained were considerably worse than when learning XML structure only. We attribute this to an over-simplification of the training data through an aggressive pre-processing step which was engaged to achieve results in a timely fashion. It remains to be demonstrated how the SOM-SD method can perform when supplied with a richer set of structural and textual information.
4.5 Conclusion

The result of our work on clustering Web like data was published in [72]. It has been shown in this chapter that the performance of a clustering task largely depends on the features of a document that was provided as input for the training process of a machine learning task. Two different tasks were considered in this investigation: one only considers the structural information of a document, and the other considers both the structure and the document content.

When the task was to deal with only the structural information of the document, a reasonable performance can already be obtained through the use of a small portion of the document structure. However, it has been observed that pre-processing procedures are likely to filter out information that may be necessary for the machine learning task to perform well. Also, it has been shown that the current unsupervised machine learning approaches that can deal with structured data has a high computational complexity. Therefore both SOM-SD and contextual SOM-SD are time-consuming to train data which contains a large number of documents, and/or contain a large dimensional input vector. The results for this segment of the INEX challenge confirmed that the approach taken in this research project is effective for this learning problem of grouping documents according to some structural property, as the SOM-SD approach out-performed other approaches and obtained the best result.

When the task was to deal with the structure and the document content, a performance worse than the structure only task was obtained. This was attributed to the minimal structure incorporated into the training data in order to allow a reasonable training duration while extra information about the document content was added. One interesting observation from this task is that even other teams failed to outperform the results produced by this research team’s SOM-SD approach that clustered using only the structural information (Micro F1 = 0.38, and Macro F1 = 0.34). This suggests that the content information may not be a useful feature for the learning task. This emphasizes that the appropriate choice of features is vital in machine learning tasks.

These findings indicate that in order to group documents effectively, there is a need to investigate the various features that can be extracted from a web document. The investigation into the various types of document features will assist in an appropriate selection of features to incorporate into the quality evaluation task.

It is recognized that the out-degree can be an issue for certain learning tasks. For example, in processing the graph depicted by the World Wide Web, the outdegree can be very large and the removal of redundancies can be a difficult task. A project which addresses this issue is under way. The general idea is to alter the processing technique. Instead of utilizing the state of each offspring as a network input, the state space of the (finite) network is used. The result of such
efforts have been reported very recently [55, 57]. Such an approach keeps the input dimension constant. This work does not form part of this thesis, as this emerging approach was not available at the time which this experiment took place, and to-date, has not been extensively tested yet.

Since the SOM-SD and CSOM-SD methods do not scale with the possible large outdegree in Web documents, and hence, this shows that these methods are not suitable as a basis for a clustering tasks of Web documents by document quality. Research on scalable Self-Organizing methods is ongoing, and hence, it may be possible to consider clustering of Web documents by quality in a future project.
Chapter 5

Document properties

5.1 Introduction

The analysis in the previous chapter showed the importance of feature selection in the grouping of documents through clustering experiments. However, Chapter 4 only investigated a number of aspects in the document feature, which is the structure within documents and the semantic of document segments. This chapter will look much more comprehensively at the variety of features that can be extracted from web documents, and investigate the possibility of these features assisting the evaluation of Web documents for quality assessment purposes.

5.2 Document structures

There are a number of structures that can be extracted from Web documents. For example, the structure which describes the relationship among documents on the Web (inter-document structure), or the structure based on how the document is written and segmented (intra-document structure). These two types of structures will be described in detail.

5.2.1 Inter-document structure

Inter-document structure refers to the relationship of documents to other documents. This relationship can be defined through the structure of a document’s URL, or the hyperlinks contained in a document.

Domain-based hierarchical structure

The URL of a page can often provide a clue to its location in the file system of a web server. Even the domain name alone would be able to indicate its position for Domain Name Service (DNS),
Figure 5.1: Illustration of domain based hierarchical structure

as indicated in the upper segment of Figure 5.1. The documents within the same domain can also be organized in a tree-like structure with the domain name as the root node of the tree, each slash ‘/’ would indicate a different level of depth, and the documents would be the leaf nodes. Different domains would be separated, and each have a tree structure, as indicated by the bounding boxes in Figure 5.1. The following URLs in the list would then map to the corresponding leaf nodes of the tree structures in Figure 5.1.


This type of structure is able to emphasize on the domain, depth, parent/child relationship, and sibling relationship of a document. Extensions can be incorporated to use the IP address as the root, and the domain names shifted down as the first-level nodes. This extension will additionally provide the analysis on the number and types of domains hosted per site.

**Link-based graph structure**

Link based structure focuses on the relationship of documents as indicated by the hyperlinks contained in a document. The link extraction and analysis process will reveal a web-like graph with many directed links among Web documents. The processing of graph can be challenging as cyclic patterns are often encountered. It is also common to observe a group of Web documents that have no connection to Web documents that are outside of the group, in which case, an
island will form. This island concept corresponds to the disconnected component of the bow-tie representation of the Web.

The documents containing the particular hyperlinks are said to contain outlinks, and the documents being linked to are said to receive inlinks. Then, the popular documents can be identified quite easily, as they are the documents with many inlinks; similarly, the link hubs can also be identified as the documents with many outlinks. The inlink and outlink information is commonly used in the ranking of web documents in current information retrieval systems (i.e. PageRank [19]). This information on link structure may be useful to identify popular web documents, but could be manipulated by the construction of a collection of documents that contain a large number of hyperlinks to a particular document without considering the quality or the relevance of the document. Rather, this is often done to illegitimately increase the ranking of a web page for reasons of exposure as a form of advertisement. As a result, to date, a correlation between the popularity of a document and its quality has not been proved.

This type of Web graph structure shows the relationship between Web documents, and reveals popular documents and hub nodes. The structure can be collapsed to avoid the large inter-domain hyperlinks commonly observed, so that only the hyperlinks to documents of a different domain are considered.

5.2.2 Intra-document structure

Intra-document structure can refer to the structure of the coding syntax, which for the case of Web document, is often based on HTML or XML tags. Both HTML and XML follow similar syntax rule, where a tag within another tag has to have a corresponding closing tag before the outer tag closes, and cannot span out of the tag it is enclosed in. This syntax rule allows the tags to be represented in a tree structure. As a result, XML and HTML naturally define a tree-like structure of a document.

The representation of XML document using its document structure has already been explored in Chapter 4. As the experiments show, although this type of structure is able to differentiate documents better than other approaches, the level of performance is still not sufficiently high. Also, this type of structure may have limited implication on the quality of web documents, as document structure or coding syntax have not been identified as quality criteria in the literature.

Intra-document structure can be derived from document properties other than the meta-language. For example, the formatting of the document can define the structure of a document. A Web document is commonly composed of text, the text is composed of sections, sections are composed of paragraphs, paragraphs are composed of sentences, itemized lists, and so on.
This again defines a tree like structure of a document. It may be possible that intra-document structure is a feature that can influence the user perception of document quality. For example a logically structured document may be perceived as being of better quality than documents which are largely unstructured.

5.3 Document features

Analyzing document structure may reveal some interesting properties of the web documents. However, there are numerous features that can also be extracted from a Web document to reveal interesting document properties. It can then be observed that most of these factors are based on the features of the actual document, in contrast to the relationship among components, where the component could refer to a web document or a segment of code within a document. The features will be categorized and explained in this section.

5.3.1 Layout-based features

Layout-based features commonly refer to the organization and the visual impact of the various components in a web page as they are displayed to users. The component organization aspect of document layout can be extracted by examining the position of an element in respect to the entire document, or in respect to a segment of the document. These features could be used to identify criteria such as the amount of information in a document, and the importance of a hyperlink or a keyword in a document.

In contrast, the other aspect of layout, visual impact, is more challenging to extract. The extractable clues which could provide an indication of the visual impact of a document include the colour, size, and the range and combination of colours and sizes of components in the document.

- Colours used in the document - Could refer to the range of colours, the hues of the colours used, or the number of different colours.
- Size of the document - Could include character count which excludes the tags, or could refer to the total document size.
- Size of a particular component - Could be with respect to the recommended display resolution, or with respect to the size of the document, which could reveal the document structure.
- Position of hyperlinks - The order of a particular hyperlink in respect to all the hyperlinks within the document, or the location of the hyperlink within the document.
• Position of keywords - Could assign different weights according to where the keyword appears. For example, a keyword appearing in the title would be considered more relevant than a keyword appearing towards the end of the body text.

• Number of non-textual components - Could include a count of the number of graphics or other multimedia content.

5.3.2 Time-based features

Time-based features are not features that can be extracted from a web document directly. These information are kept by the document creator, and is readily available on the web hosting server; however, these information are often not openly shared. The time-based features could provide indication on how up-to-date a document is.

• Last modified time - This information is provided by web hosting servers, and may be erroneous, but the majority of web servers will provide this information accurately.

• Age - This information is simply computed by subtracting the access time by the last modified time. This information can be valuable in the observation on the frequency of document update.

• Document creation time - This information would provide an insight into the life cycle of a document when coupled with the modification information. However, this information is usually not accessible.

5.3.3 Usage-based features

Usage-based features are extracted from the usage records that the web server could keep track of. This is a different interpretation of the popularity of a web page. Instead of analyzing the hyperlinks to a web page, this approach is analyzing the statistics based on the actual access to a page by web browsing users.

• The number of access to a page - This information can be collected and analyzed on a per day, per week, per month, per year basis, or simply using an accumulative total over the lifetime of the page.

• The number of download of a file - This is useful for non-text type of document. Recording the number of downloads provides an indication about the extent which the file is found useful by web users. This is perhaps even more useful when combined with the statistics on access, as the same person may repeatedly access a page, but download is mostly
only carried out per person. Therefore a combined information will provide additional information on the return rate of a user.

- The flow of traffic - This requires a continuous collection of data on a fine granularity or with the additional timestamp information. The information collected will provide a clue as to how the access to the web pages changes over time. A graph can then be plotted to show the traffic flow through the day, week, month or even year.

5.3.4 Text-based features

Text-based features focuses on the textual content of the document that web users see. The tags are filtered out in the extraction of text-based features. Text-based features could identify a number of criteria such as the amount of textual content in a document, the keywords of a document, the spelling accuracy of the document, and the grammar correctness of the document.

- Number of words in the document - Can be worked out with a simple word count.
- Number of unique words in the document - Only counts the number of unique words.
- Average word length in a document - It is possible that the content of documents which contain a large number of long words may me more difficult to comprehend, and hence, may be perceived as being of lower quality.
- Frequency of keywords in the document - This could be a simple frequency count of a word in the document, or an TF-IDF value for the occurrence of a word in the document multiplied by the inverse frequency of the word in the entire testbed.
- The average number of characters in a word - This could provide an indication on the type of writing in the document. Academic or other formal writing usually involve words with larger length; whereas writing that is closer to the spoken style would be expected to have less characters in an average.
- Spelling accuracy of content - This can be easily achieved through the use of a spell checker. Spelling accuracy reveals the amount of care that was given by the author in producing the document. It also has readability implications.
- Grammar accuracy of content - This is more difficult as current grammar checkers are far from state-of-the-art. Grammar is similar to spelling in the sense that it affects the readability of a document, and is only a problem when the amount of error is significant.
• Information about the content - This refers to the extraction of article title, author and other information specified within the body of the document content, in order to reveal the reputation of the author [106]. There is no standard for the presentation and location of these information, therefore the identification and extraction process is challenging.

5.3.5 Semantic-based features

Semantic-based features are perhaps the most challenging to identify, as it can often involve complex procedures or manual labeling. If an effective approach can be identified, the semantic features provide useful indications such as the document topic, the accuracy of information contained in the document, and the amount of focus or ambiguity in the document.

• The topic area of the document - This usually involves a combination of a number of features, such as word frequency, semantic extraction from the title, or Bag-Of-Word approaches to identify keywords within the topic that reveal the topic area.

• The accuracy of document content - Approaches utilized by existing search systems involve a panel of information experts. No algorithmic solution has been proposed to address this to-date.

• The amount of focus on a topic - This requires manual processing as well. Some approaches have been proposed in the investigation of semantics in a written document. These approaches could be extended to address the amount of focus on a topic within a document.

• The purpose of the document - This could be identified by analyzing the use of language or sentence structure combined with the topic area of the document.

• The ratios of opinion and facts - This could provide a clue as to how much bias could be contained in the document.

5.3.6 Miscellaneous features

There are other features that can be found in some web documents, but may not be identified in the majority of web pages.

• The security measures in place - This deals with the handling of data, especially data with privacy concern.
- The security of the server - This refers to how vulnerable the server is to attack, or unauthorized access.

- The goal of the document - This refers to whether the document is designed to deliver news, to allow discussion in a forum, to deliver technical knowledge, to describe product information for commercial purposes or others.

### 5.3.7 Quality-based features

Quality-based features are features that could provide an indication to the quality of a web document. All the aforementioned features can be used to provide an indication on the quality of a document, even the structures which may not seem obvious to the contribution of quality evaluation could provide some indirect quality indication. Some quality indicators may require the extraction and analysis of multiple features. The following list re-iterates previously mentioned properties and shows how these can be possible quality indicators. In fact, these quality indicators are derived with reference to literature and to the survey conducted by a research partner in Edith Cowen University in Western Australia [75], which is presented in more detail in Section 6.2.

- **Domain-based hierarchical structure** - This may be used to indicate credibility of the information, and the objectivity of the information.

- **Link-based graph structure** - This may indicate the accessibility of the information and the extent to which the information is well-supported.

- **Intra-document structure** - This may indicate the extent to which the information is well-organized or well-presented.

- **Layout-based feature** - This can indicate the information credibility, the extent to which the document can be understood and followed, the ease of navigation, the ease of locating a piece of information and the suitability of the document viewed in the Web environment.

- **Time-based feature** - This may indicate the extent to which the information is up-to-date.

- **Usage-based feature** - This may be used to indicate the availability and accessibility of the information.

- **Text-based feature** - This may indicate the amount of referencing support for the information, the extent to which the document can be understood and followed, the coverage and completeness of information, whether the document has an attributed author, and to determine whether the information contained in the document is sufficient.
● Semantic-based feature - This may indicate information correctness, information credibility, the extent to which the information is well-written, the coverage and completeness of information, the amount of focus in the information, whether the document contains inspiring or innovative information, the relevancy of information components to the overall topic, and the objectivity of the information.

● Miscellaneous feature - This may be used to indicate the security of information from malicious modification, and whether the document is free from the breach of copyright laws.

It can be seen from the above list that all different types of features can provide indication about some aspects of web page quality. Out of them, text-based and semantic-based features are able to provide indication on more quality aspects, followed by layout-based feature. It is then important to identify the quality aspects which are significant to the evaluation process of web pages, in order to extract the specific features that allow the analysis of the corresponding quality aspect.

5.4 Criteria for quality analysis

It can be seen that structures and a large number of document features can be extracted by analyzing the web documents. The different extractions reveal various properties of the document. For example, extraction of the hyperlinks reveals the popularity of documents. However, a distinction needs to be made between document properties that contribute towards the evaluation of the document quality, and those that do not.

There may be two situations that a document property could result in being considered a non-contributing property to the quality evaluation process.

● The property cannot be evaluated by an algorithm.

Some of the quality criteria are based on properties of the document that can only be evaluated through cognitive decision, and this may not be achievable through a machine-implementable algorithm. The requirement of a cognitive decision would imply that a manual processing is required. When the task is to evaluate the large number of web documents, manual processing cannot be incorporated into the quality evaluation solution, as otherwise the evaluation rate would be significantly decreased.

● The value of the property has no correlation with the quality of a document.
It is also quite possible that the properties extracted from documents have no correlation with the quality of the corresponding documents. This suggests that it would be more beneficial to not include the particular property into the quality evaluation task, since it may have an adverse effect.

- Contradictory or impossible quality criteria.

Human judgement on document quality can include contradictory criteria, or may demand properties which are not achievable. For example, a document may be seen as being of high quality the more comprehensible and accurate a subject area is covered while being of small size. This can not generally be ensured as for some subject area (such as the relativity theory) may require considerable amount of text in order to be accurate and comprehensible. Hence, when considering processes of human judgement in the assessment of document quality, then care has to be taken that the quality assessment criteria are a feasible reflection of an effective decision process.

This shows that it is vital to examine the quality criteria in great detail, so as to obtain an understanding on the feasibility of implementing an algorithm for the evaluation of the criteria, and to assess the correlation between the quality criteria and the document’s perceived quality. In order to do this, lists of quality criteria as identified through literature review can serve as a starting point of investigation. It was identified through literature review in Chapter 2 that some factors are consistently and widely considered as important quality evaluation indicator. They include the factors shown in Table 5.1, in the order of importance as observed in literature, evaluated by the frequency in which they are stated as an important quality factor. Some of the quality criteria may be non-trivial to assess by an algorithmic, machine implementable approach. For example, a machine does not normally have background information on any subject area, and hence, it can be difficult to devise an algorithm which can assess the accuracy of a document. Other features, such as timeliness are somewhat simpler to assess by looking at the time stamp that is associated with a Web document.

In order to answer more specifically the question on which criteria are used, and on how strong an individual criterion contributes to an assessment of quality, a user survey was conducted, which is described in detail in Section 6.2. From the survey results, a list of criteria can also be compiled in order of the amount of agreement among the participants. The top 20 criteria are listed in Table 5.2 sorted according to how strongly the participants consider their impact on quality. Each criteria is shown with their association to the literature identified quality factors.

As can be seen from the two lists, the literature and results of the user survey show some sim-


Table 5.1: Table describing the quality criteria commonly recognized in the literature

<table>
<thead>
<tr>
<th>Quality Factor</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>accuracy</td>
<td>the extent to which the information contained in a web page is correct and reliable</td>
</tr>
<tr>
<td>timeliness</td>
<td>the up-to-dateness of a web page</td>
</tr>
<tr>
<td>consistency</td>
<td>the extent to which the information on a web page is coherent and non-contradicting</td>
</tr>
<tr>
<td>security</td>
<td>the extent to which sensitive data is only accessible by authorised personnel</td>
</tr>
<tr>
<td>completeness</td>
<td>the extent to which the information contained on a web page is sufficient</td>
</tr>
<tr>
<td>reliability</td>
<td>the extent to which the information contained on a web page can be trusted to be correct</td>
</tr>
<tr>
<td>concise</td>
<td>the extent to which information is compactly represented without being overwhelming</td>
</tr>
<tr>
<td>understandability</td>
<td>the extent to which information can be easily comprehended and is clear without ambiguity</td>
</tr>
<tr>
<td>accessibility</td>
<td>the extent to which the information on a web page is openly available and downloadable</td>
</tr>
<tr>
<td>relevancy</td>
<td>the extent to which the information on a web page is of use to the user</td>
</tr>
</tbody>
</table>

Similarities; for example, accuracy is regarded as the most important quality factor theoretically and in practise, and seven out of the ten most important factors from the literature are also identified through the survey. The common quality factors are accuracy, timeliness, consistency, security, completeness, concise, and understandability. Furthermore, the top five quality factors in the literature can all be found in the top ten survey-based quality factors.

Analyzing the list from a different perspective shows that the two lists also display a great deal of differences. In fact, three of the important quality factors identified through the survey are not listed as the top quality factors in literature, they are italicised in Table 5.2, and they include believability, objectivity and and value-added. The order of the factors are also very different. Although accuracy is in the first position in both lists, timeliness changed from the second position in the literature list to the eighth in the survey list, which is in a much lower position than consistency (in the fourth position of the survey list) and lower than all other common factors.

This comparison of quality factors from the literature against quality factors from a users’ viewpoint provides an insight into the practicality of quality frameworks to practical applications. This also shows the importance of incorporating user input into a system that will ultimately be
Table 5.2: Table listing the quality criteria as recognized by web users in the order of their importance

<table>
<thead>
<tr>
<th>Quality criterion</th>
<th>Quality factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information that is clearly erroneous</td>
<td>accuracy</td>
</tr>
<tr>
<td>Information that is incorrect</td>
<td>accuracy</td>
</tr>
<tr>
<td>Information that seems unreliable</td>
<td>accuracy</td>
</tr>
<tr>
<td>Information that lacks credibility</td>
<td>believability</td>
</tr>
<tr>
<td>Pages that contain numerous spelling errors</td>
<td>understandability</td>
</tr>
<tr>
<td>Information that contains poor grammar</td>
<td>understandability</td>
</tr>
<tr>
<td>Poorly written information</td>
<td>understandability</td>
</tr>
<tr>
<td>Information that does not attempt sustain itself</td>
<td>believability</td>
</tr>
<tr>
<td>Information that seems disjointed and difficult to follow</td>
<td>consistency</td>
</tr>
<tr>
<td>Information that is not complete</td>
<td>completeness</td>
</tr>
<tr>
<td>Long-winded, unfocused information</td>
<td>concise</td>
</tr>
<tr>
<td>Information that is highly repetitive</td>
<td>concise</td>
</tr>
<tr>
<td>Unsecured or unprotected information</td>
<td>security</td>
</tr>
<tr>
<td>“Under construction” or “coming soon” statement</td>
<td>completeness</td>
</tr>
<tr>
<td>Information that lacks an attributed author</td>
<td>believability</td>
</tr>
<tr>
<td>Out-of-date information</td>
<td>timeliness</td>
</tr>
<tr>
<td>Information that is biased in nature</td>
<td>objectivity</td>
</tr>
<tr>
<td>Information that is difficult to read</td>
<td>believability</td>
</tr>
<tr>
<td>Too little information</td>
<td>completeness</td>
</tr>
<tr>
<td>Un-inspiring, boring information</td>
<td>value-added</td>
</tr>
</tbody>
</table>

utilized by the users, as it has shown that there are discrepancies between user perception and theoretical assumptions.

Taking and combining the quality factors from the literature and the survey, a final list of quality criteria can be developed as the following. A mapping of quality-based features with the list developed from survey responses in Table 5.2 will additionally reveal the relevant features which could be used for quality evaluation.

- Accuracy - semantic-based feature
- Consistency - semantic and layout-based feature
- Believability - link-based graph structure, domain-based feature and text-based feature
• Understandability - intra-document structure and layout-based feature

• Completeness - semantic and text-based feature

• Concise - semantic-based feature

• Timeliness - time-based feature

• Security - miscellaneous feature

• Objectivity - domain-based hierarchical structure and semantic-based feature

As the list shows, these factors correspond to the survey results, except that the order has been modified to take the quality factors from literature into consideration. The top three factors from literature move one position up, and the bottom three factors (those which were not included in the quality list) move one position down. Since value-added was already the least important quality criteria, moving one position down would result in value-added not warrant a place in the final list. These top 10 quality criteria will be the basis for identifying approaches to extract information or features for the evaluation of quality, which will be investigated in the next section.

5.5 Suitability of extracted feature for quality analysis

Throughout this chapter, the structure and features of documents have been shown to indicate certain properties. The quality criteria as identified in literature and through a user survey have also been analyzed, which allows the conversion of quality criteria into the appropriate feature extraction process to be made. A proposed approach is to use the final list of quality criteria from the previous section, and replace the type(s) of features by the information that can be extracted, in an attempt to reveal the extraction processes required. Table 5.3 shows the result of such an approach.

Table 5.3 lists the significant quality criteria and the feature extraction processes which could possibly assist in evaluating the corresponding quality criteria. Literature on the evaluation of the more detailed criteria which contribute to the concept of quality is limited, therefore the development of such a table is made possible through a combination of reference to the work by Klein [73] and the researcher’s practical knowledge and observation on web documents. Although Table 5.3 matches extraction processes which have the potential of providing an indication to the corresponding quality criteria, but the process may not always be feasible in the Web-based environment. Therefore, further investigation into the procedures and algorithms used for these extraction processes is required. It may be possible that some of the features cannot be extracted
Table 5.3: Table mapping the quality criteria with relevant extraction processes

<table>
<thead>
<tr>
<th>Quality Criteria</th>
<th>Relevant Extraction Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>The verification of content accuracy</td>
</tr>
<tr>
<td>Consistency</td>
<td>The correlation of various segments to the document topic area, The variation of size and colours used for components within the document</td>
</tr>
<tr>
<td>Believability</td>
<td>The amount of referencing material, The authority of the domain which it belongs to, The existence of an attributed author</td>
</tr>
<tr>
<td>Understandability</td>
<td>The amount of error in the coding syntax of the web page, The amount of spelling and grammatical error</td>
</tr>
<tr>
<td>Completeness</td>
<td>The coverage of the topic area, The size of the document</td>
</tr>
<tr>
<td>Concise</td>
<td>The amount of focus on the topic area</td>
</tr>
<tr>
<td>Timeliness</td>
<td>The last-modified-time of a web page</td>
</tr>
<tr>
<td>Security</td>
<td>The vulnerability of the server which the document is located on</td>
</tr>
<tr>
<td>Objectivity</td>
<td>The type of domain which the document belongs to, The amount of bias in the content</td>
</tr>
</tbody>
</table>

without manual effort, especially some of the semantic-based features. Also, through further exploration, other feature extraction processes not considered previously may be discovered, and may even be more appropriate for the task at hand. The investigation into each of these is reported in the subsections that follow.

5.5.1 Accuracy

The verification of content accuracy traditionally requires a large amount of manual efforts, and is usually carried out with a panel of information experts. Although accuracy is widely considered to be the most important quality criteria, but the difficulty of the verification process makes the evaluation process extremely challenging. In most cases, even information seekers would have difficulty differentiating the documents with accurate information from those containing inaccurate information, as the user would have to be familiar with the topic area to the extent that some facts about the topic are already known. In this case, a discrepancy between the previously obtained knowledge and the information on the web document would trigger suspicion about the accuracy of information on the web document. As a matter of fact, even manual effort cannot guarantee that the panel of information experts are sufficiently knowledgeable in most of the topic areas on the Web to make a judgement on the accuracy of the information.
An alternative approach would be to use a collection of data containing accurate information about most topics on the Web, as a reference for comparison. Then it would be possible to verify the accuracy of the information in a web document without manual involvement. The challenge however, is the selection of such a data collection, and the methods of comparison between the reference data and the target web document. As pointed out previously, such a data collection should cover a wide range of topics, even topics that has emerged due to the popularity of the Internet. More than that, it should also contain sufficient amount of data with accurate facts about those topics. The methods of comparison should also be addressed with care. The comparison method should be able to firstly, identify a page or a segment of a page from the reference collection which addresses the same topic as the web document to analyze. Secondly, extract the corresponding statements in both of the documents so that a comparison can take place. Then finally, be able to compare the statements while taking the linguistic features such as the following into consideration.

- Sentence polarity - Whether the sentence is an affirmation or a negation. For example, “the earth is not round” is a negation
- Synonym - Words with very similar meanings. For example, “listen” and “hear”.
- Polysemy - A word or phrase with multiple meanings. For example, “bank” as in a financial institution, and “bank” as in the raised land on the side of a river.
- Meronymy - A word which is a subset of another word. For example, “finger” and “hand”.

As can be seen, evaluating the accuracy of a document is not trivial at all. There are some challenges to address, no matter whether the chosen approach is manual intensive or automated. For this research, manually-intensive approaches will not be considered. The reason is that the system application is for the Web, which comprises of a gigantic amount of data, and therefore the approach which is feasible for being carried out on web documents is the automated approach. As a consequence, the challenges of reference data selection and comparison method identification should be addressed.

5.5.2 Consistency

The analysis on the degree of consistency in the web document could be approached from two different directions. Firstly, analysis can be conducted to evaluate consistency of content. The correlation of the subject area of various segments to the general topic of the document can be
computed. The segments exclude images, video or audio components, as they do not have text-based description associated to them, and the current technology is not able to detect the content topic area from these components. Instead, the segments could refer to the various paragraphs within the document content, or the hyperlinks and their surrounding text. The correlation of the subject area could be calculated by comparing the extracted keywords from the segment against the keywords for the document topic area. This would identify paragraphs or even hyperlinks that are not related to the topic area of the document, for example, advertisements.

The other approach is to identify the degree of consistency in the document layout. For this analysis, the size, position and colours used for components are extracted, and compared to those obtained from other components. If most of the components are of the same size, similar position and same hexadecimal RGB colour code, it could then be concluded that the document layout is consistent. In practical web design, this can be achieved using a style sheet. However, this does not imply that a web document with three different colours used alternatively throughout the document, and different sized components does not appear consistent. It is in fact difficult to judge the extent to which users would consider the layout to be consistent. Different users may even have different tolerance level for the layout consistency. Therefore, the subjectivity of document layout consistency may result in the evaluation being not feasible.

From this discussion, perhaps evaluating consistency of document can only be achieved through the investigation into content consistency of segments within the web document.

5.5.3 Believability

There are three possible methods for the evaluation of believability: the investigation into the amount of referencing material, the authority of the domain the document belongs to, and the existence of an attributed author. The first method deals with the extraction of referencing material. Web documents could reference using the traditional referencing style, or it could explicitly refer to documents through the incorporation of hyperlinks. A system which evaluates believability from this perspective would need to have the flexibility of being able to detect both referencing approaches. This may seem like a simple task, but the automated detection of the traditional referencing style is not always simple, as there are a number of referencing styles such as the Harvard and the Cambridge style, or the in-text referencing or the numbering style. The detection application would have to be able to detect all these styles. Whereas referencing using hyperlinks also require detection, because not all hyperlinks are used for referencing purposes. Some hyperlinks are provided for personal reasons such as linking to a friend’s web site or for financial reasons such as advertisement. The application to extract referencing material would
have to take these into consideration and identify only the legitimate referencing material.

The second method for the evaluation of believability is to evaluate the authority of the domain which the document belongs to. This is a challenging task, as although the domain of the web document can be easily identified, but the authority of the domain is difficult to define. This would require manual effort, and the manually decided authority score would have to be assigned to all accessible domains. This method is not feasible for this research project, as it involves too much manual effort.

The third method for evaluating believability is to detect the existence of an attributed author. The detection of an attributed author could be done by keyword search through the meta data. However, meta data can be manipulated, and is not commonly used, therefore another mean of detection will be required. Detecting an author name may be challenging without meta data, as it often does not have a keyword associated with it, and there is no consistency in its location. It may be before a body of text, or at the end of it, it could even be at the bottom of the document and appear detached from the actual content. Searching for a string of words beginning with capital letters may sometimes false identify the title as the author. Therefore, author detection is possible, however, a number of tests will be required for a successful detection of the attributed author for a web document.

There are three methods for the evaluation of believability for a document, however, only two are feasible. They are the investigation of the amount of referencing material in the document, and the detection of an attributed author for the content of the document.

5.5.4 Understandability

Understandability can be evaluated using two methods, identifying the amount of error in the coding syntax of the web page and evaluating the number of spelling and grammatical error. The first method of identifying error in the HTML coding syntax can be done in an automated manner, since the syntax rules for HTML can be pre-defined. However, web users may not always notice the presence of a HTML syntax error. This is because different browsers have various levels of tolerance to errors. For example, Internet Explorer is widely acknowledged to be the most accepting browser, so much that a missing closing tag could still allow the web page to be correctly displayed, and a miss-spelt colour would have the default colour “green” displayed. The amount of work is considerate compared to the impact, as the HTML code of the entire document would have to be examined for correctness in the spelling of keywords, and the number of matching opening and closing tags. All this work needs to be carried out in order to identify the number of syntax error does not seem like a good idea.
The other method includes the verification of the content’s spelling and grammar correctness. This could be done using the spelling and grammar checking software applications publicly available. The performance and the negative occurrences for the selected application would require examination to ensure that the application is able to produce a result within a reasonable time and that the negative occurrences are indeed incorrect and not false negatives.

For the evaluation of the understandability of the document, identifying the number of error in the HTML code may require too much testing, whereas the spelling and grammar correctness verification can be achieved using publicly available applications. Care should be taken to select the applications most appropriate for the task though.

### 5.5.5 Completeness

The evaluation of completeness for web documents can be achieved using two methods: analyzing the coverage of the topic area, and calculating the size of the document. There may be a number of approaches to analyze the coverage of the topic area for a web document, but the proposed approach is one of the simplest. First of all, the coverage of a topic traditionally utilizes manual effort, which this research project would not be able to adopt. It is again challenging to propose an approach which does not require manual involvement in the evaluation, as is the case for tasks that take the semantics of the content into consideration. Only one possible approach for this task without manual involvement is suggested. The automated analysis of the topic coverage requires the following procedure:

1. The topic area of the document should be defined
2. The keywords from the document should then be extracted
3. Expand the vocabulary by including words similar to the extracted keywords, and also keywords which often associate with the topic area or related topics
4. Compare the keywords from the document to the expanded vocabulary.

The comparison algorithm between document keywords and the expanded vocabulary should be carefully selected, so a score which is representative of the coverage can be computed.

The other method is to calculate the size of the document. The size of the document does not refer to the total number of bits contained in the document, rather, it refers to the amount of textual content in the document. This can be carried out by a count of the number of words.

For the evaluation of the completeness of information contained in a web document, a somewhat complex, but automated approach can be used to analyze the coverage of the topic area. A
different approach which is much simpler, is to count the number of words to get an impression of the amount of textual information in the document.

5.5.6 Concise

The evaluation of concise can be carried out by investigating the amount of focus in the document. The important concept here is to evaluate whether the information in the document is presented with clarity, and without repetition. This means that concise is not the opposite of coverage, and therefore the score does not need to have an inverse relationship. Concise is a challenging concept to evaluate for this research project as manual involvement is definitely required for this task. An automated approach which could be used to evaluate this is not yet known.

5.5.7 Timeliness

The evaluation of timeliness can be achieved by obtaining the last-modified-time information from web servers. Most web servers would have that piece of information, as that is often used to decide whether to use the cached version of the web page and save on the generation of traffic, or to download the freshly updated web page. Some people may argue that the last-modified-time may not always be correct, but this piece of information is in fact, the only time-based information that can be retrieved. If this approach is not taken, then the timeliness would have to be evaluated with a dedicated server which constantly downloads web pages, and checks the content to observe changes. This later approach is too time-consuming and resource intensive to be applicable in practical setting. Therefore, the former approach of obtaining the last-modified-time is commonly used instead.

5.5.8 Security

Security is the opposite of accessibility, as the more secure a web page is, more restrictions, and more inconvenient measures will be enforced to protect from unauthorized access and attacks. The evaluation of a web document’s security involves examining the security of its server, as vulnerable servers would allow unauthorised access and even modification to the web documents that reside on it.

The most common security breaches for web servers are unauthorised access and denial-of-service attacks. Unauthorised access is usually conducted by scanning all the ports for one or more that has a weak or no security protection, and then maintain unauthorised access to the server from one of those ports. Trojan horse is also a possibility, but since most web servers are dedicated for only that task, and does not allow personal use where Trojan horse could be
downloaded with insecure files, Trojan horse is not a common problem for web servers. The other security breach is through denial-of-service attack where the attacker floods requests to the web server, paralyzing the service, so that no legitimate requests could be served.

In order to detect the first type of security breach, a scan of the ports would need to be conducted. This is not a solution which can be carried out on a large scale, and in doing this, the machines used for this research project could be back-listed as attacker, therefore would not be recommended to incorporate into the final set of quality evaluation measures. Whereas the detection of the second type of security breach is easier, and can be incorporated. The detection of a web server under denial-of-service attack can be achieved by setting a time-out value for the requests. If the web server is able to respond within a reasonable period of time, then it can be assumed that the server is able to handle the current level of request, and therefore is not under denial-of-service attack. The challenge then, is to define “reasonable period of time” for this task. In fact, this time-out solution is often incorporated into the crawling application instead of being used separately as a quality evaluation measure.

5.5.9 Objectivity

The evaluation of the objectivity in a web document can be done through defining the type of domain which the document belongs to, or by directly examine the amount of bias in the document content. It is important to note that bias can refer to unjustified criticism or compliment, the absence of a piece of information can also be considered as bias. For example, in a scenarios where there is a company which evaluates all the laptop computers and provides this evaluation as advise for people wishing to purchase laptop computers; the company may provide better comments on laptops from a certain company due to some financial benefit it receives from that particular company. This is considered bias. If the evaluating company excludes laptops only from a particular company due to competitive reasons, then the information provided by the evaluating company can be considered biased as well.

The first method of evaluation investigates whether the document is in a domain which is expected to contain biased information or opinion can be achieved in practical settings. This is because domains can be separated according to the type of institution. The institutions can be governmental bodies, educational institutions, commercial institutions, non-profit organisations or others. These information is reflected in the Top Level Domain (TLD) of the URL [106]. Since there are a limited number of TLDs, assigning a bias score to each TLD is not a challenging task. However, the challenge lies in the formulation of the bias score. An appropriate approach to define the bias score needs to be decided.
The second method of evaluation is to directly investigate the amount of bias in the content. This may not be a feasible task due to the difficulty in the detection of bias without manual involvement. In addition, even if a process to detect bias has been found, the detection in a large document would be very time-consuming, as not only does the process have to evaluate the use of language, it also has to maintain information about the topic in order to detect the absence of a potentially important information.

As a result of the investigation, the evaluation of objectivity in a web document is expected to be better achieved by defining the type of domain which the document belongs to, and then obtaining the bias score for the particular type of domain.

### 5.6 Conclusion

In this chapter, various document properties were explored in an attempt to identify the properties that could be utilized for the evaluation of web document quality. As the previous chapter has shown how important the features used for machine learning tasks are, this chapter has identified and investigated the many possible features that can be extracted from web documents, including the inter-document structure, intra-document structure, layout-based features, time-based features, text-based features, semantic-based features, miscellaneous features and also the quality features. Few features were then selected from the possible features, depending on their potential to evaluate the quality criteria considered to be significant by the literature and by participants of a user survey. The selected features are then analyzed further with the aim of seeking feature extraction methods which could be implemented for the task. The result is a differentiation between implementable methods and methods that cannot be utilized for the task.

This is an advancement from the cognitive and vague description of quality criteria, to methods that can be interpreted into an algorithm and implemented so that an automated evaluation of document quality can be carried out. The results of the analysis described in this chapter can be considered as the preliminary exploration into the quantization of quality. These results also form the basis for the detailed quality evaluating algorithms in the next chapter.
Chapter 6

Quality evaluation

6.1 Introduction

The amount and variety of materials on the fast-growing Internet has made it a popular medium for information retrieval activities. The Internet is largely unregulated which renders its content to varying degrees of quality and reliability; as a result, retrieval of quality information from the Internet has been a research focus in recent years [16, 33, 92]. In addition, current information retrieval systems are reported to only index a minute portion of data from the World Wide Web, and the resources available to retrieve and index data from the web cannot keep up with the rate of growth of the World Wide Web, therefore it is important to ensure that the data retrieved from the web meet a level of quality standard, in order to prevent the waste of valuable resources. Manual filtering which usually delivers higher quality results should also be avoided, due to its resource intensiveness and its slow rate of identifying high quality web pages.

For this research, the target pages are web pages with a high quality; but due to the dynamic nature and lack of standards on the web, this identification of high quality web pages has proven to be a challenging task. In order to achieve such a goal, a number of questions need to be answered:

- What are the criteria which lead users to consider a document as a source of quality information? In other words, what do users want to see in a quality document? Note that in this research project, the term quality is independent of any specific topic or specific search query. Quality of documents is to be assessed on the basis of features which a web document exhibits. For example, documents which present information from a reliable source, or present information which is easily readable, etc. may be considered as being of higher quality. This paper seeks to answer the question on what such quality criteria might be as seen by the users of the Internet. The research project also seeks to answer the question
on how strongly users agree on the importance of certain criteria. This is because different
users may assess the quality of a web page on a different set of criteria.

- How to translate user perceptions into a machine understandable set of instructions? This
  is a particularly challenging task. User perceptions such as “a quality page must contain
  reliable information” can be quite challenging to implement on a computing platform.
  Often, user perceptions on a relativistic term such as “quality” can be based on other terms
  such as “reliable”. This research project suggests a number of methods which can be used
  to translate such cognitive user opinions into a machine implementable function.

- The above points can also be used to rank web documents. Which component is considered
  most important and how do others measure up? A weighting scheme needs to be decided.

- The outcome of the above mentioned items can be used directly as an implementation to
  enhance the experience with horizontal search engines. To enable the implementation of
  a vertical search engine, it is necessary to make estimations on the quality of a web page
  before it is downloaded \(^1\). How can such a measure on quality be estimated reasonably
  accurately? This research proposes a means which allows the estimation of the quality of
  a page which has not yet been downloaded but for which at least one reference (a source
  page containing a hyperlink to the page) is known.

The previous chapter, chapter 5 has shown that it is possible to extract features from web
pages for evaluation purposes, and that not all potential quality criteria can be evaluated through
an automated procedure due to the limited technology available for machine to interpret human’s
cognitive decision on quality. This chapter will investigate the basis and the approaches to prac-
tically evaluate the quality of web pages with the assistance of machine learning methods.

### 6.2 Foundation for quality assessment

Information retrieval systems are generally required to be able to retrieve information which
is relevant within a given context (or queries). The World Wide Web allows open access to
unregulated information, and hence, it is desirable for any information retrieval method on the
web to take the quality of information into account. This minimizes the effects of retrieving low
quality or irrelevant information which would be of little use to the user.

\(^1\)Here we make an important distinction: a crawler follows a link and access a web page. Before it is downloaded,
its quality is assessed using the method which we are proposing in this thesis. If it is assessed as a quality web page,
then it will be downloaded. If it is assessed as a low quality web page, it is not downloaded.
The results of most information retrieval methods are presented (directly via vertical search engines, or indirectly via horizontal search engines) to users through a user interface, and hence, the quality of these web documents should meet users’ criteria. In the past, various attempts to retrieve quality information from the web were based on generic assumptions on user’s perception of quality. These generic assumptions were often based on simple or atomic measures such as the total size of a document, its last modification date, and so on. However, no serious effort, as far as we are aware of, was made in assessing the criteria which users actually utilize in order to assess the quality of a web document, nor were serious efforts made to assess the strength of such a criterion. Here, the strength of a criterion can be interpreted as the amount of agreement between a group of users on how a specific criterion is perceived to influence the quality of a document.

In order to answer more specifically the question on which features are used, and on how strong an individual feature contributes to an assessment of quality, a user survey was conducted. The survey was run over a period of 12 months, and conducted at the Edith Cowan University, Australia, as a phased procedure [75].

A total of 132 participants were asked to address a total of 127 questions which were presented in an electronic form. Some of these questions were targeted at obtaining an overview of information seeking behavior, to develop an understanding of the users’ social background, to understand what type of information may be sought by a user, and to understand how a user’s view changes during the course of the survey. The outcome of the survey has not yet been published though it is available in the form of a PhD thesis [75].

For the purpose of the survey, the participating users all had to be currently active academics and/or postgraduate students who engage the World Wide Web regularly for the purpose of retrieving high quality information related to their research or work. It was assumed then, that the user group would have a relatively high cognitive awareness regarding how they interact with target information, and make choices about its quality. If a broader user group had been considered, this would have required the execution of an additional step, viz. the segmentation of participants into groups. Such segmentation procedures may potentially add “noise”, or a certain level of inaccuracy into the results obtained. By restricting the survey to one specific group of users, the introduction of additional “noise” into the results of the survey can be avoided.

Table 6.1 gives an overview of a number of questions relevant to our intent of finding quality features used by users\(^2\), and a summary of the corresponding user responses. In Table 6.1 we

\(^2\)This is only a relatively small subset of questions asked during the survey. There are many questions asked which are not related to our purpose and as a result they are not shown.
have sorted the responses in descending order according to the column “Greatly decreases” the perception of information quality of a visited web page or website. Note that out of a total 132 users who responded to the set of questionnaire, only 48 completed the entire set of questionnaire. Hence in our use of the data, we will only considered the responses of these 48 users, rather than the entire population of 132. The number of users responding to various questions is shown in bracket in Table 6.1. Each row should add up to 48. For example, in the first row, it is found that there were 0 user who specified with “Does not affect”; 4 users who responded by marking the entry “Marginally decrease”; and 44 users who responded by marking the entry “Greatly decreases”. Thus this table also provides some information on how often users agree with one another on whether they agree or disagree with the presented question.

There are some supplementary questions designed to find out the perception of the users on the existence of biases in the surfed web site. This portion of the user survey is shown in Table 6.2.

Based on this portion of the user survey, we have the following observations regarding the user perception of what they considered as quality web pages or websites.

- Information that is clearly erroneous or incorrect, unreliable, or lacks credibility appear to have induced users to attach highest perception on the lack of quality of the web page or website encountered.

- Generic information related to the “presentation” of web page or website, e.g. spelling accuracy, grammatical errors; that the information is written up poorly; that the information appears to be disjoint and difficult to follow; the information appears to be incomplete; the appearance of the words “Under Construction”, “Coming soon” seem to have accorded perception of low quality.

- The lack of reference or the importance of having an attributed author, appears to be a useful feature, though by no means as important as the perception of erroneous or incorrect information.

- The way the information is expressed, e.g. long windedness, repetition of information also appears to be a useful factor.

- The timeliness of the information appears to be a factor.

- The perception of bias in the information contained also appears to be a factor.

- By contrast, it appears that users are less concerned on the appearance of irrelevant information, information that appears to be out-of-place, unhelpful information, time it required...
Table 6.1: WWW information characteristics and their relationship to perceptions of quality

Indicate how your perception of information quality of a visited webpage/website changes when the following characteristics are encountered on those pages

<table>
<thead>
<tr>
<th>Information</th>
<th>Does not affect</th>
<th>Decreases Marginally</th>
<th>Decreases Greatly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information that is clearly erroneous</td>
<td>0.0% (0)</td>
<td>8.3% (4)</td>
<td>91.7% (44)</td>
</tr>
<tr>
<td>Information that is incorrect</td>
<td>0.0% (0)</td>
<td>14.6% (7)</td>
<td>85.4% (41)</td>
</tr>
<tr>
<td>Information that seems unreliable</td>
<td>0.0% (0)</td>
<td>16.7% (8)</td>
<td>83.3% (40)</td>
</tr>
<tr>
<td>Information that lacks credibility</td>
<td>2.1% (1)</td>
<td>16.7% (8)</td>
<td>81.3% (39)</td>
</tr>
<tr>
<td>Pages that contain numerous spelling errors</td>
<td>4.2% (2)</td>
<td>27.1% (13)</td>
<td>70.8% (34)</td>
</tr>
<tr>
<td>Poorly written information</td>
<td>2.1% (1)</td>
<td>27.1% (13)</td>
<td>70.8% (34)</td>
</tr>
<tr>
<td>Information that contains poor grammar</td>
<td>4.2% (2)</td>
<td>25.0% (12)</td>
<td>70.8% (34)</td>
</tr>
<tr>
<td>Information that does not attempt sustain itself (e.g.; reference etc)</td>
<td>8.3% (4)</td>
<td>31.3% (15)</td>
<td>60.4% (29)</td>
</tr>
<tr>
<td>Information that seems disjointed and difficult to follow</td>
<td>4.2% (2)</td>
<td>45.8% (22)</td>
<td>50.0% (24)</td>
</tr>
<tr>
<td>Information that is not complete</td>
<td>6.3% (3)</td>
<td>45.8% (22)</td>
<td>47.9% (23)</td>
</tr>
<tr>
<td>Un-secured/unprotected information (i.e. sensitive information that should be protected)</td>
<td>31.3% (15)</td>
<td>20.8% (10)</td>
<td>47.9% (23)</td>
</tr>
<tr>
<td>Long winded, unfocused information</td>
<td>4.2% (2)</td>
<td>47.9% (23)</td>
<td>47.9% (23)</td>
</tr>
<tr>
<td>Information that is highly repetitive</td>
<td>8.3% (4)</td>
<td>43.8% (21)</td>
<td>47.9% (23)</td>
</tr>
<tr>
<td>“Under Construction” or “Coming Soon” statements</td>
<td>22.9% (11)</td>
<td>54.2% (26)</td>
<td>33.9% (16)</td>
</tr>
<tr>
<td>Information that lacks an attributed author</td>
<td>2.1% (1)</td>
<td>54.2% (26)</td>
<td>43.8% (21)</td>
</tr>
<tr>
<td>Out-of-date information</td>
<td>4.2% (2)</td>
<td>58.3% (28)</td>
<td>37.5% (18)</td>
</tr>
<tr>
<td>Information that is difficult to read</td>
<td>20.8% (10)</td>
<td>41.7% (20)</td>
<td>37.5% (18)</td>
</tr>
<tr>
<td>Information that is bias in nature</td>
<td>10.4% (5)</td>
<td>52.1% (25)</td>
<td>37.5% (18)</td>
</tr>
<tr>
<td>Too little information</td>
<td>14.6% (7)</td>
<td>50.0% (24)</td>
<td>35.4% (17)</td>
</tr>
<tr>
<td>Un-inspired, boring information (nothing new or innovative)</td>
<td>25.0% (12)</td>
<td>39.6% (19)</td>
<td>35.4% (17)</td>
</tr>
<tr>
<td>Web Pages that are difficult to navigate</td>
<td>35.4% (17)</td>
<td>31.3% (15)</td>
<td>33.3% (16)</td>
</tr>
<tr>
<td>Information that is hard to find</td>
<td>33.3% (16)</td>
<td>35.4% (17)</td>
<td>31.3% (15)</td>
</tr>
<tr>
<td>Information that probably breaches copyright laws</td>
<td>39.6% (19)</td>
<td>29.2% (14)</td>
<td>31.3% (15)</td>
</tr>
<tr>
<td>Information aimed at the wrong audience (in the context of a website)</td>
<td>25.0% (12)</td>
<td>43.8% (21)</td>
<td>31.3% (15)</td>
</tr>
<tr>
<td>Information that seems out of place (in the context of a website)</td>
<td>14.6% (7)</td>
<td>54.2% (26)</td>
<td>31.3% (15)</td>
</tr>
<tr>
<td>Irrelevant Information</td>
<td>27.1% (13)</td>
<td>43.8% (21)</td>
<td>29.2% (14)</td>
</tr>
<tr>
<td>Unhelpful information</td>
<td>37.5% (18)</td>
<td>33.3% (16)</td>
<td>29.2% (14)</td>
</tr>
<tr>
<td>Information that is difficult to understand</td>
<td>33.3% (16)</td>
<td>41.7% (20)</td>
<td>25.0% (12)</td>
</tr>
<tr>
<td>Information that does not meet your information needs</td>
<td>56.3% (27)</td>
<td>18.8% (9)</td>
<td>25.0% (12)</td>
</tr>
<tr>
<td>Pages that contain out-of-date/broken hyperlinks</td>
<td>25.0% (12)</td>
<td>54.2% (26)</td>
<td>20.8% (10)</td>
</tr>
<tr>
<td>Content that takes a long time to download</td>
<td>60.4% (29)</td>
<td>22.9% (11)</td>
<td>16.7% (8)</td>
</tr>
<tr>
<td>Too much information</td>
<td>72.9% (35)</td>
<td>25.0% (12)</td>
<td>2.1% (1)</td>
</tr>
</tbody>
</table>

An interesting aspect, which appears to be ambiguous in the survey results, concerns too much information or too little information. It seems some users attach some importance to too little information and some attach less importance on the fact that too much information is presented.
Table 6.2: Survey questions addressing the perception of information bias distribution from the World Wide Web. Other TLDs not listed will receive a default value of 50%, indicating unbiasedness.

<table>
<thead>
<tr>
<th>URL Address</th>
<th>Response percentage</th>
<th>Response count</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.URLname.org">www.URLname.org</a></td>
<td>12.5%</td>
<td>6</td>
</tr>
<tr>
<td><a href="http://www.URLname.com">www.URLname.com</a></td>
<td>10.4%</td>
<td>5</td>
</tr>
<tr>
<td><a href="http://www.URLname.edu">www.URLname.edu</a></td>
<td>66.7%</td>
<td>32</td>
</tr>
<tr>
<td><a href="http://www.URLname.gov">www.URLname.gov</a></td>
<td>10.4%</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>URL Address</th>
<th>Response percentage</th>
<th>Response count</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.URLname.org">www.URLname.org</a></td>
<td>16.7%</td>
<td>8</td>
</tr>
<tr>
<td><a href="http://www.URLname.com">www.URLname.com</a></td>
<td>68.8%</td>
<td>33</td>
</tr>
<tr>
<td><a href="http://www.URLname.edu">www.URLname.edu</a></td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td><a href="http://www.URLname.gov">www.URLname.gov</a></td>
<td>14.6%</td>
<td>7</td>
</tr>
</tbody>
</table>

- Users appear to be aware that .com web sites might contain biased opinion, while .edu web sites might contain quality information (from Table 6.2).

The survey data provides us with some “ground truth” data which help to develop an understanding as what features help to assess the level of quality of a document on the web. One may ask: would the population size be too small to draw any useful conclusion. After all, only 48 users completed the entire questionnaire. The selection of quality features presented in this paper is guided by the results of the survey but by no means critically dependent on them. Indeed in the selection of the quality features we do not need to make use of the quantitative nature of the survey. It can be surmised that even if we have a larger population in the user survey, we may still obtain similar results, as we can expect that most people would use similar features to guide them in finding quality web pages or websites. The aggregation of the individual quality feature scores to obtain a quality score could be somewhat dependent on the survey results, in that we could use the information on the frequency of the users agreeing with one another to weigh the individual quality features. While the resulting weights could be affected by a larger population, since we are using a normalised set of weights (normalised through percentage operations), it is expected that these weights would not be too sensitive to the increase in the number of participants in the survey. As it is, in this paper, we carefully handled this possible dependency by also considering a linear combination of features with equal weights. Thus, on both counts it is anticipated that the
proposed approach in this paper is not critically dependent on the number of participants in the user survey. Obviously if we can obtain a larger population of users to participate in the survey the better.

However, judging by the results, especially those which are conducted blind in nature, in that a user is required to indicate if a particular list of returned web pages or websites are perceived as high quality or not, without knowing if the list is returned using a quality mechanism or a relevance mechanism, there is good agreement of the user expectation of quality pages and the results obtained through our approach. Hence this could serve as a post hoc validation that the population size used in the survey is sufficient to indicate the ways in which a user would use various features to guide him/her to perceive if a particular web page or website is of high quality.

In the following section (Section 6.3), details of the selection of 12 criteria are included to highlight the importance of each feature

6.3 Quality criteria

In the process of developing the possible criteria to evaluate the quality of a web page, many features were considered, but not all were included for various reasons, as was discussed in Chapter 5. One example is the metatag, since its vulnerability to abuse is widely asserted [23, 120]. As a result, a careful selection of 12 high impact quality criteria were chosen. There will be two subsets of features, one based on information about the web pages before the page content is retrieved (link-based), and the other subset which depend on the features of the web page (page-based). These are addressed in Section 6.3.1 and Section 6.3.2 respectively. Both the link-based features and the page-based features can be used to evaluate the quality of web pages in the information vault. Note that in this aspect, we use the surveyed results as a guide on to look for features which can be obtained on the links of a page and features which can be related to web pages to determine the quality of a web page, rather than using the actual survey results in providing quantitative parameters on machine implementable algorithms to the crawler. The main reason is that we wish to find ways to determine the quality of web pages independent of the content of web pages. In other words, the set of quality features is to be independent of contents, and intrinsic in nature to the web page.

These quality assessment modules can be used directly in applications such as horizontal search engines where documents are available for assessment. Horizontal search engines such as Google compute a score for each page whenever a search query is issued. While the details to how exactly such a score is computed by Google are unknown, it is known that Google utilizes a page
popularity measure based on PageRank, a relevance measure based on keyword matching, a spam measure based on link analysis, and other measures [14, 2]. The weighted sum of these produce a final score for each page. These scores for the web pages are then sorted in a descending order. Pages with the highest score are returned first by the search engine. It can be observed that a quality score measure can be quite easily incorporated in such a framework. Moreover, such a method is very efficiently computed since the quality score of a page does not change as long as the associated page and its parent pages do not change. Thus, a quality score is computed once just like the PageRank which also needs to be computed just once.

A more interesting application of the quality assessment module arises with vertical search engines (also known as focus crawlers). Here, the requirement is the computation of a quality score before a web page is retrieved. In other words, a focus crawler aims to retrieve documents which are likely to meet certain quality criteria. This in turn requires an estimation of the quality of the web page based on links pointing to the page. The following gives details of the proposed framework and presents a novel solution on how one can maximize the accuracy of the estimation of a quality score of some unknown target page.

6.3.1 Computing the Quality Score of links

This section contains details for a number of quality features that address the properties of individual links in a page (parent page) to another page (child page), based on the information readily available before the content of the child page is retrieved by downloading it. Note that these features by themselves do not allow conclusions to be drawn about the quality of a child page. This is because the overall quality of a page is assessed on a number of features including those which are computed on the actual page. A proposal on how a quality score of a child page can be estimated is given in Section 6.6.

As indicated in the following subsections, the score for individual quality feature can be computed, based on the characteristics of the hyperlinks and other features associated with the links. These individual quality scores do not depend on the content of the web page, and hence can be computed from the link information, or information which can be obtained before the web page is downloaded. Some of the features used are not commonly associated with quality assessments, e.g. location of a link within a text, timeliness of a web page or website, but are mainly used for other purposes. However, by using such features, we will show they are useful in estimating the quality of information contained in a web page. There are five such features:

- Anchor text
• Link location

• Timeliness

• Bias. We make a differentiation between positive bias and negative bias as perceived by the user on the information presented.

**Anchor text**

This component is used as an extension to many link-based algorithms such as Google’s approach of using the anchor text to provide indication about the topic area of the descendant page in addition to the pageRank[19, 25, 36, 62, 101]. However, instead of relating the anchor text to the topic area, the anchor text is used here to indicate the value of a link to a descendant page. This approach has been found by [101] to be an effective indicator for executing high-precision focus crawling.

The rationale behind this approach is to evaluate how relevant the linked page is to the parent page’s topic area. For example, if a web document discusses extensively on space exploration, and it contains hyperlinks to other web documents that also address topics related to space exploration, we could assume that those linked documents will have a high probability of being of good quality, since they are considered useful by a person that knows the topic well. However, if the same initial page also contains a hyperlink to a document that does not address a similar topic, then there is a lower probability that the particular hyperlink is included because of its usefulness and relevance to the topic. Therefore, the hyperlink is assumed to lead to a web page which the target audience could not refer to for more information on the topic.

**Link location**

This component identifies the location of the link within a web page. This criterion was not listed by user to be of relevance. However, we found that the location of a link contributes significantly towards the computation of a (estimated) score of a target page. Another reason that the link location contributes positively to the score estimation may be that it differentiates the various hyperlinks within the same web document. Since all hyperlinks within the same document would have the same score from the analysis of the parent page, only their quality score in this subsection 6.3.1 will be able to provide some differentiation in the score of a target page. It is quite possible, and was sometimes observed that the elementary score (the unweighted score of all the components) of hyperlinks within the same page are identical. In this case, it is confusing for machine learning approaches to learn that identical elementary scores are actually supposed
to result in different quality scores. This is believed to be the main reason that the addition of link location score improves the machine learning performance, as it provides differentiation in the elementary score among the hyperlinks contained within the same document.

**Timeliness**

This component determines whether the web page is sufficiently up-to-date. Timeliness is the 4th common dimensions of information quality according to literature as summarized by [76]. This component is especially important for news articles, as news that has been out-of-date is of very little value to users. Note that the time stamp is not actually a property of a link itself. It is listed here since the time stamp can be obtained by requesting a target web page without actually downloading it.

The request returns a time stamp of the target page. In practice, this time stamp is normally used to enable web browsers to use locally stored cache data if a web page has not been changed between visits. The measure of timeliness is not restricted to the comparison of current time to the last modified time of a web document. It could alternatively be the comparison between the time between the modifications, the comparison of current time and the time that the page was first created, or even other measurements. The reason for the chosen approach is because most web servers would have information about the time that the web page was last modified available in the document’s header, therefore would allow such a comparison possible in most cases. Whereas other measures of timeliness rely on information about a web page that is not always available.

**Positive bias**

This component aims to identify the probability of the web page content being biased. Users from the survey pointed out that all information contain bias; however, the probability and amount of bias vary greatly.

**Negative bias**

Negative bias has the same basis as the positive bias, except that the probability score is calculated from a different survey question that addresses the bias issue from a different perspective [75]. It should be noted that the positive and negative biases do not necessarily add up to 100% for the same gTLD.
6.3.2 Computing the Quality Score of a given document

The previous subsection addressed the scores that can be derived by analysing the limited information about the particular web page before the retrieval of its content. The following seven quality features can be applied to any given text document (i.e. no information about the context of a document is required, and thus, no information about the link structure is required). These features are:

1. Spelling accuracy,
2. Document size,
3. Existence of references,
4. Existence of authorship,
5. Non-spam probability,
6. Grammar correctness, and
7. Correctness of content.

Spelling accuracy

This component aims to identify the percentage of correctly spelt words in the document, to understand the readability of the document. Participants from the quality perception survey agreed on the importance of such a component, as 70\% of survey participants believe that pages with numerous spelling errors greatly decrease their perception of the page quality [75].

Document size

This component was introduced with the aim of identifying documents that are too short to contain sufficient information, which many survey users consider a factor that decreases their perception of the document’s quality. Document size component also directly evaluates the amount of data in a document, which is also a common dimension of information quality in the literature as summarized by [76]. It is interesting to see that 72.9\% of survey participants responded that a document with "too much information" does not decrease their perception, therefore indicating that too much information is not a sign of low quality. However, the same cannot be said about "too little information", as 50\% of participants stated that too little information marginally decreases the quality of the document, and 35.4\% of participants stated that too little information greatly decreases the quality of the document.
Reference count

This component aims to identify documents that provide referencing information to support the claims and information contained in the document. Referencing information is identified as one of the components to confirm the reliability of the page content, and reliability is listed as the 7th common dimensions of information quality in the literature according to [76]. More than a half of the survey participants also recognize the importance of references by stating that the lack of references to sustain the information greatly decreases the perception of information quality of a web page [75].

Authorship existence

Information about the author is identified to assist users to evaluate the reputability of a web page, as reputation is listed as the 17th common dimension of information quality [76]. The quality survey showed that 93.8% of participants would have a marginally or greatly decreased perception of the page quality if the page lacks an attributed author [75].

Non-spam probability

This component aims to differentiate non-spamming documents from those that could be spam. Spam has been a known issue for practical search systems. They range from the manipulation of the system so that their spam documents are ranked higher, to the inclusion of many keywords so that the spam document appears in as many search results as possible. These are means of increasing the exposure of the spam documents, but cause annoyance to users of search systems, and affects the perceived usefulness of the search system.

Grammar correctness

Grammar correctness assists in evaluating the understandability of the document, and is therefore identified as a component of understandability, which is 13th on the common dimension of information quality [76]. Survey results also show that 63.8% of participants from the quality perception survey believe that their perception of the page quality would greatly decrease if the page contains grammatical errors [75].

Content correctness

Content correctness component under the category of accuracy is the top most common dimension in the literature for information quality [76]. The importance of this component is confirmed
with 86.3% of survey participants indicating that erroneous information greatly decreases the perceived quality of a web page[75]. Thus, content correctness is considered an essential index for assessing the quality of a web page.

6.4 Approaches

The procedure to obtain the quality score involves firstly, comparing the list of common quality dimensions in the literature [76] to the results of a survey on users’ perception of quality [75], then conduct an investigation into some quality factors that could be implemented. Both of the tasks mentioned as part of the procedure were carried out in Chapter 5, which allowed the identification of a set of quality criteria which are considered important in the literature, and are relevant for Web users in assessing the quality of a document, it also allowed the investigation into the features of a web page that could indicate its quality. Resulting in a set of high impact criteria from which we selected the 12 most important ones for further analysis (as is shown in Section 6.3.1 and Section 6.3.2 respectively).

Since this thesis concerns the development of a prototype for quality information retrieval, and hence, no attempt is made to exhaustively address all known quality criteria. We will focus our attention to the developed of a quality framework that is limited to 12 high impact quality criteria. This list can be expanded to refine the performance of the system as a future exercise in this research area. The framework proposed in this thesis allows the inclusion of additional quality criteria in a straightforward manner. It will be shown in this thesis that most of the selected 12 criteria are useful to enhance user satisfaction with information retrieval systems on the Internet. Moreover, we differentiate the use of the these 12 criteria into two processes: one is to be incorporated into a vertical crawler, another is to be used as a ranking algorithm.

6.4.1 Quality assessment during crawling

Incorporating a quality assessment module into the crawling process has several benefits. Firstly, a vertical crawler would be able to retrieve web documents in an order that corresponds to a priority score. By incorporating a quality assessment module, the priority of web document retrieval will ideally reflect the quality of the web documents, and therefore documents of higher quality will be expected to be retrieved earlier in the crawling process. This allows a collection of high quality documents to be obtained efficiently, therefore attempts to crawl exhaustively can be avoided.

Secondly, the size of the Web is known to be extremely large. In fact, the Web can be seen as
the world's largest repository of digital documents. As a result, no single search engine has been able to index a significant portion of the Web. For example, it was reported that the total number of web pages indexed by at least one of the various search engines is approximately 17 billion by February 2007 [38], out of the estimated total of 447.98 billion pages on the web as was projected in Chapter 3. With a crawling process that retrieves web pages with no guidance, the quality level of the documents retrieved is expected to vary greatly, similar to what can be found on the Web. However, by incorporating a quality module into the crawler, there can be an assurance that if the same number of documents are retrieved, a large portion of the documents will be of an acceptable level of quality. These are only two of the major advantages of developing a quality module for incorporation into a crawler.

Although it is advantageous to crawl according to the quality of web documents, it has challenges as well. As mentioned earlier in this chapter, a decision needs to be made by the crawler whether to retrieve a document or not, before the content of the document is “seen” by the crawler. Therefore, in order to allow an accurate decision to be made before the retrieval of the document, an estimation or an estimation of its quality score is required. To produce an estimation of the overall quality of an unknown page, in this thesis we propose the use of a popular machine learning method viz. a multilayer perceptron (MLP). This is performed in three phases which are summarized briefly as follows:

**Phase 1:** This phase determines how well each of the quality features is suitable as a basis for an estimation, by examining the relationship of the corresponding features in the source (parent) and target (child) page pair. As foreshadowed in Section 6.3 there are altogether 12 such features. Hence there would be 12 inputs, describing all the considered potential features which could be used for score estimation, and 10 outputs, describing the actual features of the child page 
, for the MLP. An MLP is then trained to produce a mapping from input quality features (parent page) and the output quality features (child page). This is called a parent-child pair.

As illustrated in Figure 6.1, the input for learning using MLP would be \( s_1 \) to \( s_n \), where \( s_i \), \( i = 1, 2, \ldots, n \) denote the values of the quality features (both link-based and page-based) associated with the parent page, with the target output being \( f_1(P_x) \) to \( f_m(P_x) \), where \( f_i() \), \( i = 1, 2, \ldots, m \), are the functions for calculating the corresponding quality features of the child page.

\[ \text{As may be observed later, we found that some of the features are less sensitive in the determination of the quality features, and hence the number of input features is reduced to 11. In addition, two of the link-based features, namely anchor text and link location do not significantly contribute to the determination of the aggregate quality score of the child page, but are rather introduced to assist in the quality score estimation. Hence the total number of output features will be 10.} \]
supplied page, which is the child page X in this case. The other set of data would have $s_1$ to $s_n$ the same parent page to child page Y, and $f_1(P_y)$ to $f_m(P_y)$ as the target output of child page Y. For this example, there are two different parent-child mappings, one from parent A to child X, and the other from parent A to child Y.

Note that the actual numerical values used in the training of the MLP are produced from the web pages using previously identified features. Note also that $m$ is fixed to $m = 10$ as that is the number of features found to contribute significantly to quality evaluation, and for estimation purposes we begin with $n = 12$, but later reduced $n$ to $n = 11$. This was done since one of the 12 quality features considered for this project is found to provide no gain in information and hence is not used for the estimation of the quality score of the child page. As a result, only 11 features will be used for the score estimation task. Please see Section 6.6 for details.

**Phase 2:** Having identified suitable features for the estimation tasks, an MLP is trained to estimate the aggregate quality score of a child page from a given parent page.

The aggregate child page quality score is computed using the 10 features that show significance in determining the level of quality in a page. These 10 features include both page-based and link-based features, and is consistent with the dimension of the output used for MLP training in Phase 1. Two approaches were adopted to arrive at the aggregate child page quality score from the 10 feature scores, and they are based on different weighting schemes, details of which can be found in 6.5. This aggregate child page quality score serves as the single-dimension target output for the MLP training.
The 11 features found useful for the estimation task from Phase 1, include both page-based and link-based features. Page-based features will be inherited from the parent page, but link-based features can be obtained from the child page. This is because the child page link-based features can be obtained without downloading the child page at all. They are available from the hyperlink information or information which can be obtained without having downloaded the child page. This results in a parent-child quality estimation model with 11-dimensional input and 1 dimensional output as represented by the trained MLP which can then be implemented as a module in an intelligent crawler.

**Phase 3**: An ‘intelligent’ crawler is deployed which uses the trained MLP to estimate the quality of child pages.

Pages with the highest estimated quality score known to the crawler at a given time are crawled first. The result is evaluated by computing how well the estimated score corresponds with the actual score of a page (when computed after the page is downloaded). Note that once a page is downloaded, we will have access to the page-based features. Hence, in this case, we will be able to compute the actual quality score of the child page using both link-based features and page-based features, combine the 10 feature scores into an aggregate quality score, and then compare this actual quality score with the estimated quality score calculated during crawling.

These three phases are explained in more detail in Section 6.5 through to Section 6.7.

### 6.4.2 Quality assessment during ranking

Incorporating quality assessment mechanism during ranking is much easier than during crawling, as the content of documents is available, therefore no estimation is required. Due to the removal of the need to estimate, the quality scores obtained during this process corresponds to the “actual score”.

However, an appropriate weighting scheme that accurately reflects the significance of each of the features in the assessment of the document quality needs to be adopted. A suitable weighting scheme will allow the component scores to be correctly weighed during the summation process, which will result in one single, weighted quality score. This final quality score will then be used to sort the documents in descending order, and will also determine the order in which the results matching a user search query are displayed. In this research project, two weighting schemes were considered and both were investigated for their effectiveness. Details of the weight determination can be found in Section 6.5.
6.5 Weight determination

As shown in the previous sections, there are a number of quality criteria which will be considered for the quality evaluation process. These score values can be consolidated into an \( m \)-dimensional score vector. Although these score vectors from those quality criteria can be useful for quality evaluation, however, since the score vector may increase in dimensionality (if further quality criteria are added), and since one score vector needs to be maintained for each web page, the amount of information can be unmanageable high rendering the ordering or ranking of web pages a much more challenging task as well. Therefore, we proposed to produce an overall quality score, a scalar, through the building of a weighted sum of the vector elements.

The formulation of an overall quality score for a document which is representative of the overall quality of a web page requires some consideration in combining the various component scores. It is not known at this stage, how each of the component scores should be weighted to achieve a representative final quality score. As a result, two weighting schemes are considered for this research project. One is based on the ability for machine learning methods to learn the relationships among the quality criteria, and the other is based on the agreement of survey participants on the importance of the criteria for quality evaluation.

The first weighting scheme is derived on the ability for machine learning methods to learn. It aims to deliver a final weighted quality score which places more focus on the quality criteria with more consistent relationship between the parent and child pair. This weighting scheme is obtained by investigating each component, and observing the ability for MLP to learn the component scores by comparing the errors associated with each target component score. In the investigation, 11 estimated component scores were used as the input, and one of the actual component scores was set as the output (hence, all 12 criteria are considered). The training would repeat with each of the actual component scores taking turns to be the output, one at a time while the corresponding output criterion is removed from the input set. Hence, the task is to reproduce a quality score value from the remaining 11 quality score components. The errors from each actual component are compared to reveal the amount of weight to associate with each component. This revealed a MSE of approximately 0.19 from all the components, and that the errors from the components have less than 1% of difference. This implies that from a machine learning perspective, that all 12 quality criteria are similar importance towards computation of an overall score. As a result, the weights obtained from this approach indicated that each component should have equal weighting. However, this weighting scheme does not necessarily agree with the human perceived importance of the quality criteria.

The weights from the second weighting scheme are derived from the strength of user agree-
Table 6.3: Normalized weights associated with each feature, as derived from the amount of agreement in survey response

<table>
<thead>
<tr>
<th>Features</th>
<th>Normalized weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeliness</td>
<td>0.0963</td>
</tr>
<tr>
<td>Positive bias</td>
<td>0.04585</td>
</tr>
<tr>
<td>Negative bias</td>
<td>0.04585</td>
</tr>
<tr>
<td>Spelling</td>
<td>0.1248</td>
</tr>
<tr>
<td>Document size</td>
<td>0.0947</td>
</tr>
<tr>
<td>Reference</td>
<td>0.1211</td>
</tr>
<tr>
<td>Authorship</td>
<td>0.1158</td>
</tr>
<tr>
<td>Non-spam probability</td>
<td>0.0992</td>
</tr>
<tr>
<td>Grammar</td>
<td>0.1203</td>
</tr>
<tr>
<td>Content accuracy</td>
<td>0.1361</td>
</tr>
</tbody>
</table>

ments on the significance of each of the criteria for quality evaluation. This is to ensure that the features that users consider to strongly affect their perception of a web document’s quality are included, and that the amount of influences from the target component scores correspond to the importance of the features in judging the page quality. For example, if 60% of users agreed that correct spelling is an important property of quality documents, and from those, 10% of them strongly agree, then the weighted contribution of that score is $0.5 + 0.1 \times 2$. The sum is then normalized so as to ensure that the quality score is between $[0; 1]$. The resulting normalized weights for the features are as shown in Table 6.3.

Since there is no publication which investigates the weights to associate with the various criteria for the quality evaluation process, both of these weighting schemes will be considered when conducting the experiments, for comparison purposes. A more suitable weighting scheme will be selected after the results from both weighting schemes are analyzed and compared.

### 6.6 Score estimation

For the intelligent crawler to perform effectively, accurate scores need to be estimated so a decision can be made as to whether a page should be crawled, and in which order. The score estimation takes as much information as required from parent pages (page-based features) as well as the page itself before its retrieval (link-based features), to produce a final estimated score, which is computed through the following procedure at the time that a hyperlink is discovered.
Firstly, extract the page-based and link-based features, then calculate a score for each feature to obtain individual feature scores which are weighted and aggregated according to the result of the MLP training to arrive at an estimated quality score. Finally, the URL of the hyperlink is inserted into the list of URLs to crawl, which is ordered according to the estimated quality score.

These information from parent pages and the page itself are referred to as quality feature scores, and would not be assumed to have equal weights towards the score estimation. Therefore, a set of weight determination procedures is needed. What we need to do is as follows:

- For each parent and child pair, find the mapping between the quality feature scores in the parent page, and those in the child page. There are 12 quality features $x_i$, $i = 1, 2, \ldots, 12$ characterizing the parent page, seven of these, say, $x_i$, $i = 1, 2, \ldots, 7$ are page-based, and the other five, $x_i$, $i = 8, 9, \ldots, 12$ are obtained from the link information between the parent page and the child page. These five link-based features are not dependent on the information contained on the child page. There are equally 12 quality features $t_i$, $i = 1, 2, \ldots, 12$ characterizing the quality of the child page. Again seven of these quality features, $t_i$, $i = 1, 2, \ldots, 7$ are page-based, while the other five $t_i$, $i = 8, 9, \ldots, 12$ are link-based. The idea is to find a mapping between $x, x \in \mathbb{R}^{12}$ and $t, t \in \mathbb{R}^{12}$. There will be as many such mappings as there are the number of parent-child pairs in the selected network. Let us assume that there are $N$ such parent-child pairs in the network. This mapping can then be used to study the sensitivity of the features of the parent page, with respect to the estimation of the child page features. If a feature in the parent page is found to be not sensitive then this particular feature will be deleted from the parent page features. This will reduce the dimensionality of the input feature space.

- From the $N$ parent-child pair mappings we like to have a way in which the quality of the child page can be estimated from the parent page. In this case we will need a way to summarise the 12 quality features in the child page into one. There are a number of ways to summarise these quality features. One way would be to weigh them all linearly and equally so that all features share a common weight. The other method is to use the user survey results and weigh the features according to the amount of agreement among the users. In any case, we can produce an aggregate score from these 12 features associated with the child page. We will have an aggregate score representing the quality of the child page. Let us call this $t$. Thus, we will have $x(i)$, where $x(i)$ is the $i$-th parent-child pair, and $t(i)$, the corresponding aggregate quality target associated with the child page, and $i = 1, 2, \ldots, N$; $N$ is the total number of parent-child pairs.
Find a model relating the input quality features \(x(i)\) and the aggregate target score \(t(i)\), \(i = 1, 2, \ldots, N\). This will result in a model \(t = f(x)\), where \(f(\cdot)\) could be a nonlinear model.

There are a number of approaches in the literature which allow us to build the individual parent-child pair models, evaluate the sensitivity of the parent page’s features, build an estimation model of the child page’s aggregate quality score based on the parent page’s features. The approach which we have chosen is from the machine learning literature, namely a multilayer perceptron (MLP) [63]. Even in the machine learning literature, there are other competing approaches, for example, support vector machines. We have chosen MLP because it is a simple method to use and there are well established open source software which allows us to build such models. Moreover, MLP has been shown to have universal approximation property [66], in that a MLP with a single hidden layer, can approximate the nonlinear mapping (assumed to be differentiable for simplicity) underlying the input-output pair represented by the training dataset, to an arbitrary degree of accuracy, provided that there are sufficient number of hidden layer neurons. The MLP incorporates a linear model as a special case, in that if there is no hidden layer, or if it is found that the mapping does not require any hidden layers, then the MLP collapses to a linear model. This is quite a nice feature of the MLP, in that if it is found that there is no need for the use of hidden layers, then one can use the resulting linear model with good confidence.

An MLP can represent a non-linear function \(y = f(x)\) which, given some input \(x \in \mathbb{R}^n\), an \(n\)-dimensional vector, produces an output \(y\), where \(y \in \mathbb{R}^m\), an \(m\)-dimensional vector. An MLP is trained by iteratively presenting an input \(x\) which has a desired output \(t\), where \(t \in \mathbb{R}^m\), an \(m\)-dimensional vector. During training, a gradient descent method (called backpropagation or error backpropagation method [63]) is used to adjust the internal parameters called weights (a parametric representation of the function \(f(\cdot)\)) such that the output of \(f(\cdot)\) becomes close to \(t\) in the sense that the cumulative sum squared error between \(t\) and \(y\), i.e. \(E = \frac{1}{M} \sum_{i=1}^{M} (t - y)^T (t - y)\), where \(M\) is the total number of training data, is small. At the end of the training process, the MLP encapsulates a non-linear function mapping which produces for a given input \(x\) the desired output \(y\). The process is called supervised learning since a target value \(t\) is available during training to guide the adjustment of the parameters. Once trained (when the training algorithm terminates), the MLP can produce an output \(y\) for any input \(x\) even if it was not part of the training set as long as the input \(x\) is from the same domain as the training data. MLPs are known to be able to generalize (i.e. make an estimation on the target value \(t\) of unseen input data \(x\)) very well as long as the training set covers the problem domain reasonably well. In other words, a suitably trained MLP may compute a quality score of some child page from a set of quality
For the purpose of this research project, we computed the quality feature scores of a relatively small set of web pages drawn from the web. This set is then used as the training set for the MLP where the input is initially a 12-dimensional vector representing the 12 individual feature scores (both the page-based and the link-based ones) of each of the parent pages, and the output is a 12-dimensional vector representing the quality feature scores of the child page. Since the features based on anchor text and link location do not assess the quality of the child page but rather the page to which the associated links point to. These feature scores only form part of the input, and not the target. Hence the output dimension is reduced to 10. The quality feature scores of the target page is available, as the page is available. Thus, the MLP can be trained by minimizing the cumulative sum squared error $E$. Once trained, the network can be presented with the quality feature scores of parent pages, and then produces the quality feature scores of a (possibly unknown) child page as output. In other words, the MLP computes on the basis of quality feature scores of parent pages, a mapping of the quality feature scores of a child page.

We propose to approach the building of the estimation of the child page quality score in a phased approach as shown in Figure 6.2.

Phase 1 starts with a set of $n = 12$-dimensional vectors (the quality feature scores of parent pages) as input, and the associated $m = 10$-dimensional child page vectors. An MLP is trained on a set of input/output patterns for each of the parent-child pair relationship in the training data. Once training is complete (i.e. it has converged) then the MLP is applied to the same set of data in order to compute the cumulative sum of the squared error $E$. If $E$ on any of the output values
is too large (with respect to some prescribed threshold), then it is said that the associated quality feature does not sufficiently contribute to the task of estimating the quality feature scores of a page. Such a quality feature is consequently removed from the dataset which effectively reduces the input dimension \( (n) \) by 1. The training procedure is repeated with this gradual decrease of input dimension until the cumulative sum squared error \( E \) for all remaining quality features is sufficiently low. This procedure, referred to as Phase 1, determines the sensitivity of quality features which actually help to estimate the quality scores of a target page. These features are then used when commencing Phase 2.

Phase 2 is to compute an estimation of an overall quality score of a target page. The aim is to train an MLP (and hence, to produce a functional mapping between the inputs and the outputs) which can be incorporated into an intelligent crawler. During a crawl, the crawler maintains a list of links which lead to pages that have not yet been retrieved. The intelligent crawler is to arrange these links in a particular order such that pages that are retrieved first are those that are of highest quality (amongst those pages which are yet to be retrieved). This requires that the individual quality features be consolidated into a single quality score as otherwise a sorted list based on \( n \)-dimensional features is difficult to achieve. Here we propose to use a weighted sum of the \( m \)-dimensional features of the output in order to compute an overall quality score. There are two possible approaches in combining the child page’s features: one by simply combining them using equal weights, and the other one by using the user survey results as a guide on what weights to use. Nevertheless, from this process, we can obtain a single aggregate quality score associated with each child page in the training dataset.

There are \( N \) parent-child pairs in the training dataset, and each is described by a \( n \)-dimensional input vector and a single aggregate output quality score of the child, and hence it is quite simple to apply MLP method again. When the training process converges, in that either a preset maximum number of iterations have been reached, or that the cumulative squared error \( E \) is below a prescribed threshold, then a model is obtained between the input vector and the aggregate quality score of the child. This represents a mapping between the parent page features and the aggregate quality score.

Note that, here the training of MLP networks requires the setting of training parameters such as the learning rate, number of training iterations, number of parameters (weights), etc. Here we specify that we have experimented with a number of training parameters, and we are presenting results which produced the best outputs.

Note also, that the training of MLP can be time consuming. However, this needs to be performed only once. As long as the quality features do not change, the MLP does not need to
be re-trained. In other words, since one can assume that users do not substantially change their opinion on quality features within a short time frame (i.e. weeks or months) this implies that the MLP does not need to be re-trained during that time.

Once trained, the MLP can be applied very efficiently to any size of dataset [63]. The underlying complexity of such MLP is linear, and hence, this makes it a feasible module for the intelligent crawler. The bottleneck with crawlers is generally the bandwidth of the network. The CPU remains largely idle during crawling, and hence, free CPU cycles can be used to execute the function as represented by a trained MLP without having a significant effect on the speed of the crawling process.

This section described the underlying methodology of our approach. In the following, the approach is applied to real world data, and experimental results are presented.

6.7 Experimental setting and results

This section will provide more information about the practical settings, the results obtained when executing the 3 phases, and an evaluation of the performance of the focused crawler. For the machine learning tasks, a neural network simulator known as SNNS [140] is used as it is a free software that has been widely used to perform machine learning tasks for many years; therefore can be trusted to correctly perform the required machine learning tasks.

For the experiments, a snapshot of a portion of the world wide Web was taken, as part of the project on distributed crawlers. The snapshot contained 26,617,303 HTML documents containing English text from over 1,300 domains.

The experimental results are evaluated by comparing the output quality scores produced by the trained network against the target quality scores, which is measured in mean squared error (MSE). MSE is used as the measurement of performance for the experiments in this chapter, since it is one of the most widely used method for quantifying the amount by which an estimated value differs from its true value, and it assesses the variation and unbiasedness of the estimation. Such a measurement is appropriate due to the limited information available during the experiments, and its sufficient indication of the experimental effectiveness.

6.7.1 Phase 1: Scoring component evaluation

The task is to analyze the usefulness of the scoring components. A training set is produced by computing quality scores of a relatively small set of 30,635 web documents. In other words, an input/target pair was made available for the purpose of training an MLP. This size of train-
ing set was chosen since we found this to be the smallest set which produces well performing MLP networks. Using a larger training set increases the time needed to train the MLP without significantly increasing its performance.

In the following, output errors are given as mean squared errors (MSE) which are computed on the basis of the difference between network output and target value. In this phase, the network output is the quality score of the individual components, estimated based on the information from parent pages or information of the actual page that are readily available before retrieval. Whereas the target values are the quality score of individual components calculated entirely on the actual page.

A set of preliminary tests were conducted to observe the range of performance achievable, delivering an average of 0.35 MSE. This high error rate encouraged analysis on the following.

1. Overfitting:

   This is a known problem with MLP networks which can occur under certain situations. It basically means that an MLP focuses too strongly on producing given target values thus effectively loosing the capability to generalize over unseen data. We utilized a test dataset to establish whether overfitting occurred in our experiments, but no over-fitting was observed.

2. Performance on individual input training patterns:

   The error rates of patterns were sorted in descending order and plotted. Figure 6.3 shows that there are a small number of pages in the high and low ends, but most of the patterns
3. Performance on individual quality components:

The MSE of each individual of the 12 components was analysed. This is plotted in Figure 6.4 and Figure 6.5 respectively (we use two figures to avoid cluttering of the plots).

Note that errors from 2 components “anchor text” and “link location” were not plotted. This is because the scores of the actual page do not require these 2 components, therefore a comparison between the components’ training results and target output cannot be made.

The plots reveal that “authorship existence” is producing a significant error, resulting in
more than 40% of error for approximately half of the patterns. Although “Document size” and “content correctness” are shown as the next difficult components to learn, their error rates remain much lower than the “authorship existence” component, and are approximately down to 20% by the half-way mark, which is considered an acceptable level. All other components have relatively low error rates with the majority of patterns (90% of them) under 10% error.

From this analysis, it was decided that the “authorship existence” component does not help the estimation of a score for target pages and was therefore removed from the list of suitable components.

While the components were being reviewed for their usefulness to this task, extra attention was given to the “link location” component, as it does not have as strong theoretical foundation as the other components. Upon investigation, “link location” was found to improve the performance by approximately 2%. The component differentiates the input scores, so it decreases the likelihood of multiple identical inputs with different output targets, which confused the learning process. Perhaps this is the reason for the increase in performance after the addition of the “link location” component. This finding serves as a confirmation of the usefulness of “link location”, and supports the decision to include the component. As a result of this cycle of experiments, the number of component to be used as score estimating inputs is set to 11.

6.7.2 Phase 2: Achievable performance for score estimation

After the components have been scrutinized for their usefulness towards score estimation, they have to be trained to collectively estimate the actual quality score of the web page. The input for training is the score from each of the resultant 11 components from Phase 1, and the target output is the quality score of actual pages, created by taking the weighted sum of target component scores, calculated from analyzing the actual web pages.

To arrive at a single weighted actual quality score, two weighting schemes were considered, one was based on the ability for MLP to learn the component scores, and the other is derived from the amount of agreement among survey users, as described in Section 6.5. After the weighting scheme for the target score has been set, it was time to identify the settings and influencing weights for the source component scores that produce the most accurate estimation on the weighted actual quality score.

The experiments in this subsection is performed on 55,466 training patterns. These training patterns are a subset of interconnected web pages from web snapshot 3 previously described in Section 3.4, and their target values are the quality scores calculated based on the properties and
the content of each page. Another non-intersecting subset of interconnected web pages of similar size, also from snapshot 3, is used as the testing patterns. Standard back propagation, resilient propagation and batch back propagation were considered with various combinations of training parameters using the SNNS package[140], in order to identify the setting that performs the best. The preliminary results obtained showed that batch back propagation performs poorly, therefore only the results from standard back propagation and resilience propagation are compared and displayed in the following tables. Each of the following tables are divided into two segments, the higher segments show the performance obtained using no hidden layer with the recommended values for the training parameters, followed by the best performing parameter values and the medium results obtained through them. Although only the initial parameters used and the best performing parameters are displayed in the table, various combinations using other values for the training parameters were also trialed during the experiments. For example, Standard Back-Propagation was tested using learning rates ranging from 0.1 to 0.6, and the maximum tolerable error trialed include 0.0, 0.01, 0.02, 0.1 and 0.2. Whereas resilience propagation was tested using various combinations which included a starting update value of 0.1 and 0.2, the maximum update value of 30, 40 and 50, and the weight decay of $10^{-6}$, $10^{-5}$ and $10^{-4}$. The lower segments of the tables show the performance obtained using the best training parameters identified in the higher segment, but with network configurations which include hidden layers, and different number of neurons in the hidden layers.

As can be seen from Table 6.4 for the weighting scheme based on the ability for MLP to learn, the best performance of 0.00114 MSE was achieved using standard back propagation as the learning function, with a learning rate of 0.2 and a maximum tolerable error of 0.1. The network configuration was set to having no hidden layers. Resilience propagation appears to be out-performed by standard back propagation consistently. It can be seen that for standard back propagation, although the performance does improve with the introduction of hidden layer, and with increasing number of neurons, the improvement is not significant, as the MSE only improved by 0.0001. The addition of a second hidden layer does not further improve the high performance already obtained with 1 hidden layer in standard back propagation. Therefore, it was decided to utilize the weights from the best performing standard back propagation training with no hidden layer for the estimation of a final quality score.

As can be seen in Table 6.5, for the weighting scheme based on agreement from participants of a user survey, the best performance of 0.00134 MSE using no hidden layer was achieved using again standard back propagation as the learning function with a learning rate of 0.2 and a maximum tolerable error of 0.1. Only one result from the resilience propagation algorithm was
<table>
<thead>
<tr>
<th>Learning algorithm</th>
<th>Training parameters</th>
<th>Hidden layer</th>
<th>Performance (MSE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.0</td>
<td>none</td>
<td>after 100 cycles = 0.00208 after 200 cycles = 0.00269 after 300 cycles = 0.00268</td>
</tr>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.1</td>
<td>none</td>
<td>after 100 cycles = 0.00116 after 200 cycles = 0.00118 after 300 cycles = 0.00114</td>
</tr>
<tr>
<td>Resilience Propagation</td>
<td>initial update value = 0.2, max update value = 30 weight decay = (10^{-5})</td>
<td>none</td>
<td>after 100 cycles = 0.00265 after 200 cycles = 0.00261 after 300 cycles = 0.00260</td>
</tr>
<tr>
<td>Resilience Propagation</td>
<td>initial update value = 0.1, max update value = 50 weight decay = (10^{-4})</td>
<td>none</td>
<td>after 100 cycles = 0.00262 after 200 cycles = 0.00260 after 200 cycles = 0.00259</td>
</tr>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.1</td>
<td>1 hidden layer with 2 neurons</td>
<td>after 100 cycles = 0.00110 after 200 cycles = 0.00109 after 300 cycles = 0.00108</td>
</tr>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.1</td>
<td>1 hidden layer with 3 neurons</td>
<td>after 100 cycles = 0.00108 after 200 cycles = 0.00104 after 300 cycles = 0.00104</td>
</tr>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.1</td>
<td>1 hidden layer with 4 neurons</td>
<td>after 100 cycles = 0.00107 after 200 cycles = 0.00106 after 300 cycles = 0.00104</td>
</tr>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.1</td>
<td>1 hidden layer with 5 neurons</td>
<td>after 100 cycles = 0.00110 after 200 cycles = 0.00107 after 300 cycles = 0.00104</td>
</tr>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.1</td>
<td>2 hidden layers: with 5 neurons in 1 and 2 neurons in 2</td>
<td>after 100 cycles = 0.00109 after 200 cycles = 0.00108 after 300 cycles = 0.00106</td>
</tr>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.1</td>
<td>2 hidden layers: with 5 neurons in 1 and 3 neurons in 2</td>
<td>after 100 cycles = 0.00109 after 200 cycles = 0.00106 after 300 cycles = 0.00104</td>
</tr>
<tr>
<td>Resilience Propagation</td>
<td>initial update value = 0.1, max update value = 50 weight decay = (10^{-4})</td>
<td>1 hidden layer with 2 neurons</td>
<td>after 100 cycles = 0.00259 after 200 cycles = 0.00251 after 300 cycles = 0.00250</td>
</tr>
<tr>
<td>Resilience Propagation</td>
<td>initial update value = 0.1, max update value = 50 weight decay = (10^{-4})</td>
<td>1 hidden layer with 3 neurons</td>
<td>after 100 cycles = 0.00256 after 200 cycles = 0.00249 after 300 cycles = 0.00248</td>
</tr>
<tr>
<td>Resilience Propagation</td>
<td>initial update value = 0.1, max update value = 50 weight decay = (10^{-4})</td>
<td>1 hidden layer with 4 neurons</td>
<td>after 100 cycles = 0.00253 after 200 cycles = 0.00251 after 300 cycles = 0.00249</td>
</tr>
<tr>
<td>Resilience Propagation</td>
<td>initial update value = 0.1, max update value = 50 weight decay = (10^{-4})</td>
<td>1 hidden layer with 5 neurons</td>
<td>after 100 cycles = 0.00257 after 200 cycles = 0.00248 after 300 cycles = 0.00245</td>
</tr>
<tr>
<td>Resilience Propagation</td>
<td>initial update value = 0.1, max update value = 50 weight decay = (10^{-4})</td>
<td>2 hidden layers: with 5 neurons in 1 and 2 neurons in 2</td>
<td>after 100 cycles = 0.00251 after 200 cycles = 0.00241 after 300 cycles = 0.00239</td>
</tr>
<tr>
<td>Resilience Propagation</td>
<td>initial update value = 0.1, max update value = 50 weight decay = (10^{-4})</td>
<td>2 hidden layers: with 5 neurons in 1 and 3 neurons in 2</td>
<td>after 100 cycles = 0.00249 after 200 cycles = 0.00243 after 300 cycles = 0.00241</td>
</tr>
<tr>
<td>Learning algorithm</td>
<td>Training parameters</td>
<td>Hidden layer</td>
<td>Performance (MSE)</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------</td>
<td>--------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.0</td>
<td>none</td>
<td>after 100 cycles = 0.00281, after 200 cycles = 0.00279, after 300 cycles = 0.00280</td>
</tr>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.1</td>
<td>none</td>
<td>after 100 cycles = 0.00134, after 200 cycles = 0.00134, after 300 cycles = 0.00134</td>
</tr>
<tr>
<td>Resilience Propagation</td>
<td>initial update value = 0.2, max update value = 30 weight decay = 10^{-5}</td>
<td>none</td>
<td>after 100 cycles = 0.00272, after 200 cycles = 0.00270, after 300 cycles = 0.00269</td>
</tr>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.1</td>
<td>1 hidden layer with 2 neurons</td>
<td>after 100 cycles = 0.00124, after 200 cycles = 0.00122, after 300 cycles = 0.00121</td>
</tr>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.1</td>
<td>1 hidden layer with 3 neurons</td>
<td>after 100 cycles = 0.00125, after 200 cycles = 0.00122, after 300 cycles = 0.00117</td>
</tr>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.1</td>
<td>1 hidden layer with 4 neurons</td>
<td>after 100 cycles = 0.00125, after 200 cycles = 0.00122, after 300 cycles = 0.00117</td>
</tr>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.1</td>
<td>1 hidden layer with 5 neurons</td>
<td>after 100 cycles = 0.00124, after 200 cycles = 0.00118, after 300 cycles = 0.00117</td>
</tr>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.1</td>
<td>2 hidden layers: with 5 neurons in 1 and 2 neurons in 2</td>
<td>after 100 cycles = 0.00121, after 200 cycles = 0.00119, after 300 cycles = 0.00119</td>
</tr>
<tr>
<td>Standard Back Propagation</td>
<td>learning rate = 0.2, max tolerable error = 0.1</td>
<td>2 hidden layers: with 5 neurons in 1 and 3 neurons in 2</td>
<td>after 100 cycles = 0.00124, after 200 cycles = 0.00119, after 300 cycles = 0.00118</td>
</tr>
<tr>
<td>Resilience Propagation</td>
<td>initial update value = 0.1, max update value = 50 weight decay = 10^{-4}</td>
<td>1 hidden layer with 2 neurons</td>
<td>after 100 cycles = 0.00268, after 200 cycles = 0.00263, after 300 cycles = 0.00261</td>
</tr>
<tr>
<td>Resilience Propagation</td>
<td>initial update value = 0.1, max update value = 50 weight decay = 10^{-4}</td>
<td>1 hidden layer with 3 neurons</td>
<td>after 100 cycles = 0.00267, after 200 cycles = 0.00263, after 300 cycles = 0.00262</td>
</tr>
<tr>
<td>Resilience Propagation</td>
<td>initial update value = 0.1, max update value = 50 weight decay = 10^{-4}</td>
<td>1 hidden layer with 4 neurons</td>
<td>after 100 cycles = 0.00259, after 200 cycles = 0.00256, after 300 cycles = 0.00255</td>
</tr>
<tr>
<td>Resilience Propagation</td>
<td>initial update value = 0.1, max update value = 50 weight decay = 10^{-4}</td>
<td>1 hidden layer with 5 neurons</td>
<td>after 100 cycles = 0.00262, after 200 cycles = 0.00258, after 300 cycles = 0.00255</td>
</tr>
<tr>
<td>Resilience Propagation</td>
<td>initial update value = 0.1, max update value = 50 weight decay = 10^{-4}</td>
<td>2 hidden layers: with 5 neurons in 1 and 2 neurons in 2</td>
<td>after 100 cycles = 0.00260, after 200 cycles = 0.00251, after 300 cycles = 0.00248</td>
</tr>
<tr>
<td>Resilience Propagation</td>
<td>initial update value = 0.1, max update value = 50 weight decay = 10^{-4}</td>
<td>2 hidden layers: with 5 neurons in 1 and 3 neurons in 2</td>
<td>after 100 cycles = 0.00264, after 200 cycles = 0.00255, after 300 cycles = 0.00252</td>
</tr>
</tbody>
</table>
displayed in Table 6.5, because the recommended training parameter provided the least amount of MSE. A similar observation to the previous weighting scheme was made about the increasing number of neurons in the hidden layer improving the performance. This time, the performance improvement is slightly more significant, at 0.00017 MSE, when using a single hidden layer with 4 neurons. This weighting scheme based on the amount of agreement in the user survey seems to be more challenging for MLP to learn, as there is a need for a hidden layer, which results in a non-linear relationship between the features and the weighted quality score. Also, the lowest MSE observed of 0.00117 is still higher than what can be observed with the previous weighting scheme, where the lowest MSE of 0.00104 was obtained.

After the network was trained using the best performing setting, the weights for the individual components were obtained and implemented into the focused crawler. Due to the non-linearity of the input and output, the weighting scheme will not be listed. The aim is to ensure that the quality score used to set crawling priorities are estimated as accurately to web pages’ actual evaluated quality score as possible. The performance obtained from both weighting schemes in the form of MSE appear promising; however, the practical performance can only be verified once the result from the focus crawler using the set of component weights is compared to the quality score of the actual pages.

6.7.3 Phase 3: Retrieval of quality information

The quality prediction appears to perform well in theory, but the performance also needs to be verified in a practical setting to observe the amount of improvement. This will provide an indication as to whether the performance improvement is significantly better than the general improvement observable with most metrics. In order to observe the practical performance level, the weights from the best theoretical performance in the previous section were incorporated into a focus crawler. During the execution of the crawler, the feature scores, weighted by the parameters obtained from the Phase 2, are totalled to produce a single score for each page, and pages are sorted and prioritized according to this score. To analyze the performance, the estimated score of each retrieved web page is recorded in the order that they are retrieved. The result shown in Figure 6.6 presents a snapshot in time after crawling 32,000 Web documents. It should be noted that only the first 32,000 scores are shown to avoid cluttering.

It can be observed that the crawler is indeed predominantly retrieving pages which produce a higher quality score. The sudden drops in score values refer to tunnelling taking place (i.e. retrieving pages of relatively low quality score in order to reach pages of higher quality score). The quality scores drop with the continuation of the crawling process, indicating that most of the
high quality pages have already been retrieved.

The tunnelling problem is a known issue associated with focused crawlers [130]. The term “tunnelling” refers to a process of finding a path between two distinct clusters of web pages which both meet a given criterion. In other words, it is possible that there is no direct link from a group of web pages which meet a search criterion to another group of web pages meeting the same search criterion.

The literature provides only few solutions towards efficient tunnelling. For example, [130] proposed to probe the neighborhood around a known group of pages by crawling pages which are not more distant than $n$ links \(^4\) from the group of pages, then select the most promising direction based on an assessment of the neighborhood. It is shown in [130] that the approach is effective if two disjoint groups of relevant pages are not more distant than a distance of 3 links.

This research project implicitly addresses the tunnelling (through pages of low quality) through a procedure which estimates the score of pages, and hence, the best direction for the intelligent crawler can be determined without the requirement to probe around a group of known pages. Also, it can be observed from Figure 6.6 that most documents with a high estimated quality score of say above 0.8 are retrieved within the first one-third of the retrieved data. It can be seen then that web pages with high estimated scores are indeed retrieved efficiently during the early stages of the crawling process, showing that the focus crawler works as expected.

Nevertheless, attempts to minimize the effect of tunnelling were made. One suspicion was that due to the nature of crawling where URLs are only known after the hyperlink pointing

\(^4\)The \(n\) link here means that the links which are contained in the \(n\)-th neighborhood of the node of interest.
to the document is discovered in another document, tunnelling may have occurred due to the limited number of un-retrieved URLs that the crawler is aware of. Such a condition could occur at domain boundary, as it has been observed that web sites usually contain a large portion of hyperlinks to documents within the same site. Therefore, the crawler may only be able to crawl the lower scored documents of a particular site that were not yet crawled, and the documents being crawled may contain hyperlinks that point to documents of the same web site again, which would either have been crawled already, or are the lower scored document still to be crawled. This pool of lowly scored un-retrieved documents will only improve when hyperlinks to new domains or new web sites are discovered. Figure 6.7 emphasizes the domain boundaries issue by colour-coding the documents that belong to the same site.

![Figure 6.7: Scores of web pages in crawling order with colour-coded sites](image)

In an attempt to overcome this possible issue of domain boundary, the priority of documents from a newly discovered web site has been set to receive a boost in the subsequent experiment, resulting in a performance as illustrated in Figure 6.8. The boosting strategy means that the first page of a newly discovered web site will receive the highest priority, so that it can be the next page to retrieve, regardless of its calculated priority from its estimated quality score. This assists the crawler to obtain information about various web sites early-on in the crawling process, therefore avoiding the issue of domain boundaries preventing the early discovery of highly scored pages.

Figure 6.8 shows the crawling result after using the boosting approach to minimize the effect of domain boundaries. The comparison of Figure 6.7 and Figure 6.8 allows the following 2 observations. Firstly, tunnelling is still observed after adopting the boosting approach, but the highly scored pages are given an opportunity to be retrieved earlier. For example, the highly scored
Figure 6.8: Scores of web pages in crawling order with the boosting strategy

pages from the domain www.redcross.org begin being retrieved at approximately the 4700th entry without using the boosting strategy as seen in Figure 6.7, but was being retrieved earlier at approximately the 4000th entry after adopting the boosting strategy as shown in Figure 6.8. Also, most of the highly scored pages from the domain www.warmuseum.ca were retrieved by the 15100th entry using the boosting approach in Figure 6.8 compared to a delayed 15800th entry without the boosting approach in Figure 6.7. The second observation is that the boosting strategy also decreased the duration of early tunnelling. The first incidence of tunnelling shows this effect. The tunnelling occurrence of crawling pages with a score of 0.5 or less can be observed between approximately the 200th and the 4700th entry in the original performance shown in Figure 6.7. However, the corresponding tunnelling occurrence is located between approximately the 200th and the 4000th entry in the boosting enhanced performance shown in Figure 6.8. As can be seen, the duration of the early tunnelling occurrence is decreased by approximately 700 entries. From these observations, boosting strategy can be considered useful, and is therefore adopted as a permanent feature in the focus crawler.

6.8 Conclusion

The novel quality evaluation approach detailed in this chapter is based on the findings of a targeted user survey on the perception of web page quality, and some also has support from the literature. The findings allow the calculation of an evaluated quality score for each web page, which is a weighted sum of a number of quality components, where the weights are derived either from the ability for MLP to learn the individual component scores, or from the amount of agree-
ment on the importance of individual components according to participating users. Through a 3 phase experimental procedure, it was made possible to

1. Identify quality components that are useful for making a score estimation.

2. Obtain weights for each of the quality components to arrive at a estimated final score that is as close to the actual evaluated quality score as possible.

3. Observe the performance of the quality-based score estimation through focus crawling.

The experimental results from the 3 experimental phases proved that the significant quality criteria pointed out by survey participants and the literature can be interpreted into algorithms, which allowed their incorporation into a focus crawler for automated quality score estimation. After the incorporation of the quality estimation feature, the focus crawler works as expected by retrieving the known web pages according to the estimated score. Although the tunnelling effect commonly observed in focus crawling continue to be observed during the quality-based focus crawling, however, as indicated in the experimental results, most highly scored web pages are retrieved in the early stages of crawling. In addition, a simple boosting strategy was able to minimize the effect of tunnelling caused by the deficiency of known web pages at the boundary of domains.

This chapter provided the reasoning for the selection of quality features considered for the quality evaluation process, and investigated the interpretation of these significant quality criteria into an algorithm, based on features which can be extracted from web pages. This chapter also showed how the estimation of quality score is achieved. However, the description of the focus crawler adopted for the quality information retrieval has not been provided, nor has the exact algorithm which allow the component score calculation been discussed. The details on the focus crawling application and the component score algorithm will be required in order to show how it is possible to integrate information retrieval with quality evaluation. These details on the overall practical quality information retrieval system will be provided in Chapter 7.

In addition, this chapter evaluated the effectiveness of the quality prediction methods through a comparison with the quality assessment methods. No attempt is made to verify whether the quality of the retrieved documents do indeed reflect the human perception of document quality. This issue is also addressed in Chapter 7 which incorporates un-biased human judgement into the assessment procedure.
Chapter 7

Quality Information Retrieval

7.1 Introduction

The overall aim of this research project is to develop a prototype for quality information retrieval. However, up to this point, the approaches for information retrieval and quality evaluation have been discussed individually. The previous chapter, Chapter 6 is especially focused on the proposed approach for quality evaluation, and provided the foundation and rationale for such an approach to estimate and evaluate the quality of web pages. This chapter will detail the proposed approach to integrate both the information retrieval and the quality evaluation components of this research project, and how the overall goal is realized in practice. The realization of a quality information retrieval prototype is achieved through the following two tasks:

- Retrieval of quality information from the World Wide Web. This is achieved through an “intelligent” crawler in the form of a focus crawler, which has a set of criteria embedded which will enable it to retrieve and download web pages of “quality”. Note that this does not use any domain specific knowledge related to the encountered web pages. This will provide a different visibility to that provided by a crawler which uses a different criterion.

- Once a set of web pages has been accumulated in the information vault, it is possible to rank the web pages according to some criterion, e.g. popularity as typified by search engines like Google, relevance as typified by search engines like Yahoo, Nutch, or quality. In this research project, we have focused on the ranking of web pages using a quality criterion, and compared the quality ranking with those provided by a relevance based search engine and a popularity based search engine.

In this chapter, we will also report on experimentation of information retrieval which deploys a set of criteria which evaluates the “quality” of web pages, and also report on the comparison
of the performance of such a quality criteria based system against another information retrieval system that is based on a different criteria.

7.2 Implementation rationale

As indicated in Chapter 6, we propose to use a focused crawler which will guide the crawling process. To appreciate the proposed approach, a comparison with the traditional crawler is essential.

The traditional crawler would crawl the World Wide Web, and download all the pages associated with the links on the page, then rank the web pages according to some criterion (i.e. by popularity as in [19]) into the information vault. The intention is that once the Web is crawled to a reasonable extend, the downloaded web pages can be indexed and reverse indexed, and can be retrieved from the data store accordingly. However, if it is possible to provide some “intelligence” to the crawler, it will only crawl and download web pages which satisfy a certain set of prescribed criteria. One would ask: why bother to provide the crawler with any “intelligence” at all. We can simply download all pages encountered like what happens in a traditional crawler.

However, in this case the web visibility is limited to what is provided by the crawler. If we embed a crawler with a certain set of prescribed criteria, then the intelligent crawler is capable of reaching out to the portion of the which might not be visible to the common crawler. In other words, the visibility of the World Wide Web is dependent on the crawler. If we provide it with a set of prescribed criteria, for example quality, it can provide a different view of the World Wide Web compared with that which is provided with say a popularity criterion (like a common crawler), or other sets of prescribed criteria, such as relevance.

Obviously, if the crawlers, intelligent or otherwise, are used to retrieve information from the World Wide Web and assuming that the retrieval speed is infinitely fast, and that we can run the crawlers over extended periods of time, then, there would not be much difference to the visible web provided by different types of crawlers. However, as it is, the retrieval speed of web pages is finite, even with the resources available to corporations like Google or Yahoo, which deploy many parallel crawlers continuously crawling the World Wide Web. Thus, there will be some differences to the visibility of the Web as provided by different types of crawlers. This is demonstrated in [34] through empirical tests that indicate very little over-lap among the search systems.

One may argue that although traditional crawler would not be able to offer the functionality of crawling prioritization, the prioritization of web page retrieval is possible through a focus
crawler. However, the proposed approach is still different from the standard “focus crawler” [130] in that the standard focus crawler will concentrate on searching information pertaining to a given topic, while the intelligent crawler crawls for information based on a set of prescribed quality criteria and is not dependent on a particular topic. Intelligent crawlers use some real-time scoring mechanisms to make a decision on whether a page should be retrieved; the scoring mechanisms aim to guide the retrieval of web pages along more promising directions, e.g. quality.

One may ask: in this case, what types of criteria would be suitable, which form the basis of an intelligent crawler in its decision to download or not. There are a number of possible criteria sets, for instance, we could have a set of criteria which pertains to evaluating the quality of the web page, or we could have a set of criteria which pertains to evaluating the relevance, or reliability of the web pages. Whether it is possible to formulate such set of criteria is a different matter. But at least, conceptually, it would be possible to have a set of criteria pertaining to evaluating the quality or the reliability of web pages. If we have such a set of criteria, then the visible Web for this crawler could be quite different to that provided by one which uses a popularity criterion.

Focus crawlers are often used in digital library applications for which it is desired to retrieve web pages addressing a pre-determined topic (see for example [40, 90, 130]). Our major focus is on the possibility of retrieving quality information from the World Wide Web. Hence in our approach, the crawler will be embedded with a set of features which pertains to characterize the quality of web pages. A conceptual framework for our approach would be as follows:

1. Find a set of features in which users use to guide them to determine if an encountered web page is perceived to be of high quality or not. There are two occasions in which such a set of quality features will be used: to rank the web pages already downloaded according to their quality and to rank outgoing links from the current page, so that the crawler can decide on the priority of links to follow (before the web pages are downloaded). Ideally we will have two sets of such features (bearing in mind how we would like to use the features in Steps (3) and (5) respectively); one set for predicting the quality of the encountered web page based on the hyperlink information alone, or information which provides some measure of the quality of the encountered links without having to download the page at all, called link-based features (to be used in Step (3)); and the other set is used to determine the quality of web pages in the local information vault, called page-based features (to be used in Step (5)). The page-based features will be dependent on the characteristics of the web page.

2. Build a prediction model which is capable of predicting the quality of a child page based
on the characteristics of the parent page’s page-based features, and the link-based features of the child page. We will call this the parent-child quality prediction model.

3. Embed the parent-child quality prediction model into a crawler, so that it will predict the quality score of the encountered web page. If the encountered web page is of high quality then it will be downloaded into the information vault, and the hyperlinks contained in the downloaded web page be followed using the same quality prediction mechanism. If the encountered web page is deemed to be of low quality, then it will not be downloaded, and the crawler will follow hyperlinks in other seed pages.

4. After certain time has elapsed, there will be a set of web pages accumulated in the information vault. This set of web pages can be indexed and reversed indexed in the normal fashion, see e.g. [19].

5. The web pages in the information vault can be ranked using a number of possible criteria: popularity, relevance, quality. The ranking of the popularity of web pages can be performed using the PageRank method [19]. The ranking of web pages according to their relevance can be performed using, say, a version of the probabilistic method as advocated in [118, 119]. The ranking of web pages according to their quality can be determined using the page-based and link-based features which can be used on web pages.

6. The ranked pages, using a user interface, can be presented in descending ranked order to the user in response to a user query.

7. If the user clicks at a particular web page link returned in the user interface, the user will be directed to the live web page on the World Wide Web and retrieve the information.

The following sections will provide the technical details on how quality information retrieval is achieved through the implementation of a focus crawler with quality estimation feature, and a quality-based ranking algorithm.

7.3 Focus crawler with quality estimation feature

For the Quality Information Retrieval system proposed in this research project, a specifically designed crawling application would be required to support the functionalities. A suitable crawling application should be able to be directed by a quality score in order to retrieve web documents that are above a quality threshold. The crawling application would also need to determine such a quality score in real-time to the crawling process. In addition, the crawling application should be
able to carry out the quality score calculation and the required crawling in a reasonably efficient manner.

During the course of the research project, a crawler has been developed to construct testbeds for information retrieval experiments, as is described in Chapter 3. The crawler is a distributed, depth-first crawler with a central coordinator. Although the crawler was able to retrieve web data efficiently, it cannot support the quality evaluation tasks. Therefore, the depth-first slave component of the crawler was replaced by a focus crawling component, which could be directed by a quality score. This allows the crawling application to continue performing efficiently under a distributed environment, yet still able to prioritize crawling according to the evaluated quality score.

In this section, a new approach to distributed focus crawling is suggested. Once again, there are two components in the proposed architecture: the "slave" is the stand-alone focus crawling component, and the "master" is the central slave coordinating component, making this a centralized distributed crawling system. Since there is no significant modification to the "master" coordinating program, the following subsections will only provide discussion on the stand-alone focus crawling component.

7.3.1 The stand-alone crawling component

As mentioned in Chapter 3, the purpose of the crawling component is to crawl a portion of the web from a given set of starting (seed) pages as a stand-alone crawler and provide regular feedback to the coordinating program. The stand-alone crawler is a focus crawler that selectively retrieves data from the World Wide Web, as opposed to a horizontal crawler that retrieves all web pages without prioritising the crawling order or filtering unimportant web pages out of the list of candidate URLs to crawl. The process in the crawler that determines what web page to crawl next is based on a scoring mechanism that assigns a priority score to web pages. During crawling, the crawling component refers to the priority score to make decision on whether and when a page should be retrieved, and thus, aim to guide the retrieval of web pages along the more promising direction. This scoring mechanism is based on some quality criteria. This section will describe the implementation of a focus crawler which is able to be guided by quality criteria, and is able to be executed both alone as a single process or under a distributed environment.

This stand-alone focus crawling component is based on the basic breadth-first stand-alone crawling component described in chapter 3. Therefore, similar to the former “slave” component, the design of the crawling component incorporated features that allow efficient and effective interaction with web pages as well as the central coordinating program. These features include
Figure 7.1: Diagram showing the 5 major components in the crawling slave

the following.

- Minimal communication overhead by only obtaining an initial set of seeds pages from the
central node, which is usually small. Crawling is then able to begin, in which new web
pages are discovered and can be retrieved.

- Compressed data transfer to minimize impact on the network. This is carried out if data
is to be collected at the central coordinating node, otherwise data resides at the crawling
component’s machine and no data transfer is carried out.

- Failure recovery procedures

- Links within comment tags or java-scripts are not followed by the crawler

- Dynamic pages that do not physically exist are not crawled to avoid crawling traps

- DNS caching on central node

Since the features in the focus crawler in respect to its interaction with the central coordi-
nating component is consistent with the former depth-first crawler developed, these features will
not be described in detail again. Rather, the focus for this section will be on the structure of the
focus crawler and its unique functionalities of determining a quality score for each web page in
real-time, and prioritizing crawling according to the quality score.

**Structure of the focus crawler**

The stand alone crawler has five major components. The relationships of the five components are
as illustrated in Figure 7.1, and the details of each of the components are as follows.
1. The administration node

The administration node contains the crawling list handler, but interfaces with the content retriever and link extractor elements.

The administration node is responsible for managing all other components as well as ensuring that the crawling list handler is only accessed by at most one source at any given time. The independent crawling component is a multi-threaded application, where both the content retriever and the link extractor modules are in the form of threads, so that multiple instances of them can be executed simultaneously. The simultaneous execution of threads is useful to minimize the wastage of time associated with different time requirements of modules for different web pages, as there is often a discrepancy between server response time and link processing time due to the unpredictable amount of network traffic and the various document size of web pages. Because of the decision to enable multi-threaded crawling, the administration node is important. The procedure of tasks carried out by the administration node is as follows.

The administration node firstly obtains an URL from the crawling list by contacting the crawling list handler with an empty string. After obtaining the URL, the crawling manager sleeps/idles for a predetermined period of time, which is calculated based on the number of threads in execution at the time, this is done in a similar manner to the breadth-first crawling component described in Chapter 3.

Secondly, the crawling manager requests the content retriever to retrieve the content of the given URL. The administration node has a time-out setting on the content retriever, in case the server that hosts the particular page traps the retriever by keeping the retriever waiting indefinitely. The administration node also keeps count of the failed retrieval attempts, where eight consecutive failures imply that the hosting server may be down, in which case the domain would be crawled again after a period of time.

Finally, the administration node initiates a link extractor and passes the page content to it. The manager allows up to eight simultaneous link extraction threads, since the time requirement for link extraction is dependent on the size of the web document and the number of hyper-links. If link extraction threads reach the maximum of eight, the administration node pauses crawling until the number of link extraction threads decrease. Otherwise, the process is repeated by executing from the first step again, until there is no URL left to crawl and that all link extractor are no longer active.

While the loop is being performed, the crawling list handler within the administration node
module may be contacted by the link extractor when links are discovered. To avoid simultaneous modification to the crawling list from multiple threads, the crawling list handler is synchronised to only allow access from one source, at most.

2. A crawling list handler

The synchronized crawling list is a sorted linked list which is maintained by a list handler within the crawling manager. It stores the links discovered by the link extractor and their corresponding scores. Links that do not belong to the site currently being crawled are stored as external links. Internal links are checked against URLs already crawled, and are discarded if matched. The unmatched links are then compared to the crawling list. If an entry already exists in the list, the entry is removed and re-inserted in the correct location of the linked list after updating the score. All unmatched internal links are inserted into the crawling list in the appropriate location according to their score.

3. The content retriever

The content retriever retrieves the content of a given URL from the World Wide Web. Several settings are in place to ensure error-free retrieval. Those include disabling redirection and pop-up pages, and restricting crawling depth in a loop dependent approach. This is similar to the HTTP input stream in the crawler detailed in chapter 3, therefore will not be revisited.

4. The link extractor

The link extractor module in this focus-crawling application is a merged module of the link extractor and URL filter modules of the breadth-first crawler in Chapter 3. It performs the search and extraction of hyperlinks from the content of a web page, as well as the formatting and filtering of the URL string. It was decided to merge the two modules when the basic crawler described in Chapter 3 was being extended into a focus crawler. This is done because the two tasks can be seen as a single process; in other words, if a hyperlink is to be filtered out due to it being dynamically created by a script, the link extractor simply would not return the URL to the administration node. This is more efficient than the previous approach of having a separate module to process and filter each URL found as a hyperlink, which would require a large number of instance creation, and would have to return a default string if the URL is to be filtered out.

5. A score calculator

Score calculator is an independent module that was developed in an attempt to quantify the findings from information quality investigation. This module performs quality score calcu-
lation in real-time. As the final quality score comprises of several component scores, this module stores scores from each quality component into an array, so that they can be easily manipulated to form a final score that can be associated with an URL. The separation of individual component scores also allows score weighting to be performed in the final score calculation process. This has provided flexibility in the manipulation of weights when a different weighting scheme is chosen. Details about the information quality investigation are included in Chapter 6, and the algorithmic implementation for each quality component is included in the next section.

The significant features of this focus crawler which allow some “intelligence” are the score calculator and the crawling list handler. Note that the score calculator calculates the component scores of current page, and the limited information about the links contained within the current page. So that the unique links within the current page that has not yet been retrieved, would each have an array of the elementary component scores. These component scores are based on 11 criteria, and they are grouped into link-based features and page-based features. The following subsections show the algorithmic interpretations of both the link-based criteria and the page-based, which are used in the focus crawler to arrive at the array of elementary scores for each link.

### 7.3.2 Calculation of link-based component score

The link-based quality features are a selection of features which can be applied to the hyperlink in a page (parent page) leading to another page (child page), based on the limited information available about the child page before its retrieval. These individual quality scores do not depend on the content of the web page, and hence can be computed from the link information, or information which can be obtained before the web page is downloaded. Note that these features by themselves do not allow conclusions to be drawn about the quality of a child page. This is because the overall quality of a page is assessed on a number of features including those which are computed on the actual page.

These four criteria however, assist in the estimation of the quality score greatly, therefore all four of the link-based criteria are incorporated into the focus crawling to enable an accurate quality score estimation when combined with the page-based criteria in refsub:focuspg. A proposal on how a score of a child page can be estimated is given in Chapter 6. There are five such features:

- Anchor text
- Link location
- Timeliness
- Bias. We make a differentiation between positive bias and negative bias as perceived by the user on the information presented.

**Anchor text**

The score of this component is dependent on the degree of relevance of the anchor text with respect to the content of the document. The score is calculated using the following steps:

1. Identify a link and its anchor text
2. Extract the keywords of the anchor text and the document
3. Score = the frequency of anchor keywords in the document.

In cases where no anchor text could be detected, the default value 0 is assigned. Anchor scores are normalized by capping at a maximum of 100 and then divide the score by 100 to produce a score ranging from 0 to 1.

**Link location**

This feature is useful in differentiating the links on the same parent page, and therefore assists in the score prediction. The score for this feature is calculated using the following steps:

1. Identify a link and the amount of text after the link
2. Extract the text that follows the particular link
3. Score = amount of text after the link/total document size. The score lies in the range of [0, 1].

It is unclear why the location of a link relative to the entire page should be a consideration. One reason may be that web page authors consciously or subconsciously refer to quality links early in the web page rather than later in the page, similar to the position of query terms in a query string. The assumption is that the terms positioned earlier in the total query string is of more importance than those positioned later in the query string, the similar assumption perhaps may be applied to the position of hyperlinks in a document.
**Timeliness**

This feature determines whether the web page is sufficiently up-to-date. The score for this feature is determined by the last-modified time stamp of the web page, returned by its server, as it is an indication of how up-to-date the information may be. In cases where the last-modified time stamp is not available, a default value is assigned.

The request returns a time stamp of the child page. In practice, this time stamp is normally used to enable web browsers to use locally stored cache data if a web page has not been changed between visits. The timestamp is converted into the number of seconds since epoch (1 January, 1970) and taken away from the time that the page is retrieved (in seconds). If the difference is more than a prescribed threshold, which is the number of seconds in 10 years \((p = 10 \times 365.25 \times 24 \times 60 \times 60)\), it is set to 0. This is because a document modified 10 years ago or more do not make too much difference in their out-of-datedness, but instead, will reduce the significance of smaller values if they are not capped. The time of last modification is then normalized by dividing by the threshold of the number of seconds within 10 years.

The steps to calculate the timeliness score are:

1. Obtain the current time in seconds since epoch \((c)\)
2. Obtain the last modified time in seconds since epoch \((l)\)
3. Compute the difference \((d) = c - l\); threshold \((p) = 10 \times 365.25 \times 24 \times 60 \times 60\)
4. Score \(s = \begin{cases} 0 & \text{if } d \geq p \\ 1 - (d/p) & \text{if } d < p \end{cases}\). The score lies between \([0, 1]\).

The measure of timeliness is not restricted to the comparison of current time to the last modified time of a web document. It could alternatively be the comparison between the time between the modifications, the comparison of current time and the time that the page was first created, or other measurements. The reason for the chosen approach is because most web servers would have information about the time that the web page was last modified available in the document’s header, therefore this would allow such a comparison to be possible in most cases. Whereas other measures of timeliness rely on information about a web page that is not always available.

**Bias**

The score for this feature is calculated by identifying the gTLD of the URL, then assigning a bias probability score based on the positive influence of bias directly derived from the survey result.
This gives the positive bias score. For example, if we find that the top level directory is a .edu site, then the positive bias score is 0.667.

The probability score for a negative bias is calculated from a survey question that is worded differently so that users consider bias from a different perspective [75]. For example, from Table 6.2, if the top level directory is a .edu site, then the negative bias score will be 0. All unlisted top level domains will be given a bias score of 0.5, indicating unbiasedness.

Since the positive and the negative biases are computed from different methods, it is reasonable to assume that they are independent features. Hence in our approach, both positive and negative biases are considered as independent and hence they are weighted differently in obtaining a composite quality score.

7.3.3 Calculation of page-based component score

The page-based quality features are a selection of features which can be applied to a given page where the page content is available for analysis. For the case of using within the focus crawler, the page where content is available would be the current page, or the parent page of a link. For the score estimation that is required to perform during focus crawling, only criteria that assist in the score estimation are considered. As a result, only six out of a total of seven page-based criteria are used. The page-based features assessed during focus crawling include the following.

1. Spelling accuracy,
2. Document size,
3. Existence of references,
4. Non-spam probability,
5. Grammar correctness, and
6. Correctness of content.

Methods which return a score between 0 and 1 are described for each of the seven criteria, where 0 indicates that the web page does not match the criterion and 1 means that it perfectly matches the criterion.

Spelling accuracy

A score can be quite easily computed by the use of general-purpose spell checkers, but there is the possibility that special terminologies or uncommon proper nouns are incorrectly labelled as
mis-spelled. To quantify the spell accuracy of web pages, we used Aspell, a publicly available and widely used spell checker [7], which includes a feature that allows the addition of special terminologies into its dictionary. The score for this component is the percentage of correctly spelled words in the web page, relative to the total number of words in the web page. Thus, a value of 1 indicates that all words in the web page are spelled correctly. Note that some web pages do not contain any text but are a composition of images, multi-media content, or others. Such web pages generally receive the perfect score of 1.

The steps taken to calculating the score for this feature are as follows:

1. Filter out the HTML tags
2. Identify unique words and count the number of unique words in the web page ($n$)
3. Pass the unique words to Aspell[7] and obtain the number of correctly spelled words ($c$)
4. Score = $c/n$

**Document size**

This feature was introduced with the aim of identifying web pages that are too short to contain sufficient information, which many survey users consider a factor that decreases their perception of the web page’s quality. It is interesting to observe that 72.9% of survey participants responded that a web page with "too much information” does not decrease their perception, therefore indicating that too much information is not a sign of low quality. However, the same cannot be said about "too little information”, as 50% of participants stated that too little information marginally decreases the quality of the document, and 35.4% of participants stated that too little information greatly decreases the quality of the document.

This score is also used as a reference for other features’ score calculation. A score is computed simply by counting the number of words in the document, and thus, web pages which contain no text receive a score of 0. The score is dependent on the amount of textual information, where more text results in a higher score, but the score will no longer increase after reaching a prescribed maximum threshold.

The score for this component is calculated using the following steps:

1. Filter out HTML tags
2. Identify and count properly formed words ($w$), where properly formed words are words without numerical values and non-punctuation symbols
3. Score = \[
\begin{align*}
&\frac{w}{800} & \text{for } w < 800 \\
&1 & \text{for } w \geq 800
\end{align*}
\]

The score is kept within a fixed range where web pages of more than 800 words are capped and set to a maximum score. This is determined through some preliminary experiments conducted before the large survey that showed that the upper limit of 800 words is a reasonable length for a web document to contain sufficient information. This capping of the score is performed to ensure that the variation within the range of 0-800 would have a significant impact. There is no penalty for documents which are of more than 800 words. Although some may question the quality of unreasonably large web pages, there is no negative scoring for pages with unusually large amount of text, since survey users do not consider too much information to be a factor that decreases their perception of a page[75].

An alternative approach could be taken where the web page size refers to the total size of the HTML document, including the HTML tags. Since users do not usually view the source code of a web page while browsing, therefore the total size of the source code would not correspond to the document size as observed by the users. This is the reason that the alternative approach was not considered for evaluating the document size.

**Reference count**

This feature aims to identify web pages that provide referencing information to support the claims and information contained in the web page. This feature attempts to accommodate different referencing styles, and the score is dependent on the number of references provided. The score for this feature is calculated using the following steps:

1. If found keywords “bibliography” or “references” towards the end of the document, score = 1

2. Otherwise, count the total number of links in the web page (t)

3. Extract the bottom one-third of the web page

4. Count the number of links in the bottom segment (s)

5. Score s = \[
\begin{align*}
&\frac{s}{10} & \text{if } s/t \geq 0.5 \\
&0 & \text{for } s/t < 0.5
\end{align*}
\]

As the procedure shows, the score is computed by searching for the keywords “bibliography” or “references” towards the bottom of the web page. If these keywords are not found, then the
hyperlinks are checked to investigate if more than a half of the hyperlinks appear at the bottom of the web page. This is because of the fact that most web pages often refer to other web pages through links which are located towards the bottom of the page. If neither keywords nor the links at the bottom of the page are detected, the score is set to a default value of 0. Otherwise, the score is simply the total number of references divided by 10, with the maximum number of references capped at 10.

The identification of referencing information in a web page has been widely agreed to be of high importance, however, there is currently no straight-forward approach of accurately quantify such information. The identification of reference is especially challenging in a web page, as hyperlinks point to supporting web pages only some of the time, with no requirement to do so, or any standard to differentiate referencing links from others. Therefore, the current approach of estimation based on the location and quantity of hyperlinks is a possible attempt made to identify references in a web document.

**Non-spam probability**

This feature aims to differentiate non-spamming web pages from those that could be considered as spam pages. The score for this feature is the probability of the web page being a spam by calculating the average word length in the header. According to the paper [98], if the average word length is more than 8, it has a 50% chance of being a spam page.

The score for this feature is calculated using the following steps:

1. Extract only the header section of the document
2. Filter out the HTML tags
3. Work out the average length of the words \( k \) in the header section
4. Score = \[ \begin{cases} 0.5 & \text{if } k > 8 \\ 1 & \text{if } k \leq 8 \end{cases} \]

Some commercial search engines implement a number of heuristics in an attempt to identify spam, however, those heuristics mostly cannot be applied for this quality evaluation purpose. The reasons being that most of the heuristics are topic specific, and since this quality evaluation algorithm does not take relevance of the page into consideration, it is not possible to incorporate them. Also, most commercial search engines combine a number of heuristics to target spam pages that attempt to manipulate their scoring system, and therefore, those heuristics are not appropriate for this quality evaluation. As a result, this generic spam detection method by [98] is adopted.
Grammar correctness

Grammar correctness assists in evaluating the understandability of the web page. The score for this feature is based on the number of grammatical errors or ambiguities identified by the Queequeg grammar checker [115], in relation to the web page size.

This score is calculated using the following steps:

1. Filter out the HTML tags
2. Pass the content to a grammar checker (we use the Queequeg grammar checker in our experiments)
3. Score = (length score (w)-number of grammatical errors and ambiguity warnings (g)/average word length(k))/length score (w)

Queequeg grammar checker is not the most accurate grammar checker currently available, but it is the only open source grammar checker that can be executed through command lines. It should be noted that identifying a suitable grammar checking application was not trivial. As most grammar checkers are not designed for Unix-based operating system, or were designed as a plug-in for text processing software, instead of an independent application that could be invoked using command lines. In addition, grammar checking is a difficult task to perform that even the most sophisticated grammar checking software would miss an error, or falsely indicate an error. Therefore, the amount of error is modified using the average word length due to the fact that a preliminary test indicated that web pages that address a field in a more specific and in-depth approach, usually use longer words and more complex sentence structures, which the grammar checker would warn against. Therefore, the inclusion of average word length ensures that the more scientific or formal documents are not incorrectly scored.

Content correctness

Content correctness is considered an essential feature for assessing the quality of a web page. But how would one design an algorithm to assess the correctness of the content of a web page? This can be a challenging task even for information experts. As in most cases, the user would have to be knowledgeable in the topic area in order to verify the degree of accuracy for a given web page.

We propose to use a trusted source of information to help assessing the correctness of a given page. Here we propose the use of Wikipedia as a trusted source, as it is currently the largest free and open online encyclopedia, covering a huge range of topics over 1.8 million articles as of July
2007 [68]. The Wikipedia content is updated by the community collaboratively and regularly, therefore the articles may not be of uniform quality [67]. Nevertheless, it was found that the quality of scientific entries are comparable to an actual encyclopedia such as Encyclopedia Britannica [48]. Therefore, it can be assumed that the information that is presented in the Wikipedia is information on which most users (experts in the field) agree that it is correct.

The Wikipedia dataset used for this analysis of content correctness is the openly distributed collection of wikipedia documents from Wikimedia Foundation in 2007, which is 12GB in size and consists of 5,456,651 documents[135]. Approximately 32.02% of the documents are redirection pages[135], which means that the actual number of documents used as reference for content correctness is 3,709,277. For every document in the Wikipedia dataset, a word frequency vector is produced by using the well-known Bag of Words (BoW) approach [88]. This needs to be performed just once for each document in the Wikipedia dataset. We assess the correctness of the content of a particular web page by comparing its word frequency vector in the BoW approach with the best matching word frequency vector in the Wikipedia dataset. The greater the similarity, the higher the score. Note that this may seem a rather crude approach. However, it turns out that this works quite well, judging from the results obtained as indicated in Sections 6.7 and 7.5.

In addition, we are currently looking at the possibility of taking word context into account in making the comparisons. This is performed by producing a context graph for each document, then to compare this with the best matching graphs obtained from the Wikipedia dataset. There is an efficient machine learning method called Graph Self-Organizing Map which can perform this task in linear time. However, this is beyond the scope of this research and shall not be addressed further in this thesis.

The approach taken to score this feature uses the dataset of Wikipedia[135] as a reference, and follows the following procedure:

1. Filter out the HTML tags
2. Identify the topic area of the document
3. Identify a reference page from the Wikipedia dataset [135] that addresses the same topic area with the assistance of the Lucene Java libraries [5, 84].
4. Compare the document with the corresponding page in the Wikipedia dataset [135]
5. The score is dependent on the degree of similarity of the document with those in the Wikipedia dataset.
7.3.4 Observations

The above mentioned approaches allow a total of 11 features to be considered when obtaining the elementary component scores. These individual component scores are then multiplied by the estimation weights, which were identified through the MLP training as discussed in Chapter 6, to arrive at a final estimated score for those links. The estimated quality score is essential to assist the focus crawler in determining whether the links contained in the current page should be crawled, and in which order.

The quality scores of those linked pages are only estimated scores as they are unseen pages which only their URL and the content of their parent page are known. It has been shown in the experiments of the previous chapter in Section 6.7 that the weights used for arriving at the estimated quality score performs well with a mean square error of 0.00114, which is small enough to be neglectable. The URL of each linked page and its estimated score are then submitted to the crawling list handler. The crawling list handler then enters the URL in the correct location on the sorted linked list according to its score. If the URL already exists on the crawling list with a different score, the score for the URL will be adjusted to reflect its average score, and moved to the correct location. This incorporation of an estimated quality score into a focus crawler is one application of the quality evaluation mechanism. The other application is to use the quality evaluation mechanism in ranking the search results to it is explained in the next section.

7.4 Quality-based ranking algorithm

The previous section described a focus crawler which is able to retrieve web pages that are estimated to meet a criterion, and in this case, the criterion is a quality threshold. For this application of the quality evaluation mechanism, no estimation is required, as ranking is usually performed once web pages have been retrieved and stored in an information vault, and therefore the content of each web page is available for analysis to allow the calculation of their quality scores.

This quality score is again based on several component scores. However, some of the link-based features were incorporated in the focus crawling score estimation and some of the page-based features were excluded to assist the quality score estimation of an unseen page. Since all necessary page contents are available for the ranking process, no estimation is required. It follows then that some of the features that do not have direct indication of the page quality can be excluded, and the one with indication of the page quality but did not assist in score estimation can be included. The result is a careful selection of 9 high impact quality criteria, chosen based on the analysis of the literature as well as a user survey as discussed in the previous chapter.
section will describe the algorithmic interpretations of the criteria used to produce component scores, which have not yet been discussed up to this point.

These criteria are categorized into two groups. One group, called link-based features, are selection of features that can be extracted without analysing the page content; and the other group called page-based features, which lists a selection of features which can be applied to the content of a given page. Note that while most features are guided by the results of the user survey, there are some features, which are found to be useful in the literature, but were not included in the user survey, are also included.

Methods which return a score between 0 and 1 are described for each of the criteria, where 0 indicates that the web page does not match the criterion and 1 means that it perfectly matches the criterion.

### 7.4.1 Computing the quality score of links

This section contains description for the quality features that can be extracted without analyzing the content of a web page. Since estimation is not required during the ranking process, only two out of a total of four link-based criteria are assessed, because of their relevance to the quality of a web document. These two features are:

- Timeliness
- Bias. Again, we make a differentiation between positive bias and negative bias as perceived by the user on the information presented.

Since the calculation procedures and algorithms for these two link-based quality criteria have been discussed previously in 7.3.2, they will not be discussed here again.

### 7.4.2 Computing the quality score of a given document

The following seven quality features can be applied to any given text document (i.e. no information about the context of a document is required, and thus, no information about the link structure is required). Since this assessment are to be carried out during the ranking process, all seven of the page-based criteria can be included. These features are:

1. Spelling accuracy,
2. Document size,
3. Existence of references,
4. Existence of authorship,
5. Non-spam probability,
6. Grammar correctness, and
7. Correctness of content.

Most of these criteria have been discussed previously in 7.3.3, therefore their score calculation method will not be repeated here, except for one of the criteria - existence of authorship, which was not included as a quality criteria during focus crawling because of its negative impact on the estimation of the quality score. Therefore, its score calculation procedure will be provided below.

**Authorship information**

Information about the author is identified to assist users to evaluate the reputability of a web page. This feature was not incorporated in the focus crawler, as it was observed that this feature does not assist in the estimation performance of the MLP training. This could be a result of the challenge experienced in locating the authorship information on a web page, due to the lack of standards. Some web pages contain the author’s name at the top of the page, some at the bottom of the page, and they are usually provided within the body of the document content without any identifiable keyword.

Since the score calculation and implementation procedures were not discussed previously, details about it are provided here. The score for this component is calculated using the following steps:

1. If found author metatag, score is 1, otherwise continue with the following steps
2. Extract only the body of the document
3. Filter out the HTML tags
4. Search in the first two lines and the last few lines for signs of name (2 or 3 Consecutive words with capital letters)
5. Score $s = \begin{cases} 
1 & \text{if author seems to be provided} \\
0 & \text{if no evidence that author is provided} 
\end{cases}$

This approach seems rather simplistic, but this feature has been one of the most challenging one to convert into a machine implementable function. No other approach for identifying authors in web documents is currently known, as far as the research team is aware.
7.4.3 Observations

As observed, the conversion of the features of web pages into quality scores could be quite challenging. This is especially the case in converting a feature like “correctness of the web page”. Often such a feature is geared more to human evaluation of the given web page or website. In our proposed approach we have used a particular method to evaluate this. There are doubtless other ways in which the “correctness of a web page” can be evaluated.

Using the proposed set of quality feature set, it will be possible to compute the quality scores of each feature of a given web page. We will need to mimic what human does in combining the individual feature scores into a composite score using various weighting schemes. The identification of the various weights could be quite challenging, as unfortunately even we humans cannot articulate explicitly what relative weights that we used to evaluate the quality of a web page. All we know is that if we look at a particular web page, we will be able to say if this web page is a high quality one or a low quality one. It would be very difficult for us to probe into our own psyche to say, e.g. that we use a weight of 0.5 for the “correctness of document” feature, and 0.1 for the “correct spelling” feature. In the previous chapter, we have considered 2 different weighting schemes, in the following section, we will continue to present the proposed ways in which weights can be found to combine the scores of various features to an aggregate one. We will also observe the performance of the overall quality information retrieval system.

7.5 Experimental setting and results

This section will provide more information about the practical settings, and an evaluation of the performance of the intelligent crawler and the quality information retrieval system. The experiments are carried out on a snapshot of a portion of the World Wide Web which was taken as part of the project on distributed crawlers, as described in Chapter 3. The snapshot contained 26,617,303 HTML documents containing English text from over 1,300 domains.

7.5.1 Focus crawling performance

Before conducting the experiment using the newly developed focus crawler, a weighting scheme needs to be decided. As discussed in Chapter 6, two different weighting schemes were considered. The first weighting scheme is based on the ability for MLP to learn each of the component scores, and the second is derived from the amount of agreement in the user survey. Both of these weighting schemes were explored under the experimental settings.
MLP-based weighting scheme

The MLP-based weighting scheme resulted in equal weights associating with all quality component scores. Discussion of this weighting scheme can be found in Section 6.5. Based on this weighting scheme, a final quality score can be calculated by simply adding all the component scores calculated on the child page. This final quality score is used as the output of the MLP network, and the elementary component scores calculated from the parent page without accessing the child page, are used as the input for the MLP network. After the network was trained using the best performing setting as discussed in Section 6.7, the estimation weights for the individual components were obtained and implemented into the focus crawler. Due to the fact that no hidden layer was involved in the machine learning process for the selected network configuration, the process of incorporating the estimation weight was straightforward.

During the execution of the focused crawler, the component scores, weighted by the estimation values from the machine training results, are totalled to produce a single score for each page, and pages are sorted according to this score in the linked list during the crawl. It was already observed in chapter 6 that web pages with high estimated scores are retrieved efficiently during the early stages of the crawling process, therefore showing that the focused crawler works as hoped for.

The aim for this experiment though is actually one step further and to ensure that the quality score used to set crawling priorities are estimated as accurately to web pages’ actual evaluated quality score as possible. This is to verify that the focus crawler is correctly retrieving the web pages that match or exceed the quality threshold. The performance of 0.00114 MSE during learning appears promising, as that will be the best performance achievable in practise. However, the practical performance can only be verified once the result from the focused crawler using the set of component weights is compared to the quality score of the actual pages. Therefore, a list of crawled URLs and their actual quality score is maintained so that when the scores are normalized into percentage, the web pages above a normalized score of 70% in the actual score can be identified as high quality web pages. The 70% quality threshold is selected so that the highly scored web pages labeled as high quality is approximately 10% of the total number of documents. A graph is then plotted to compare the efficiency of retrieving high quality pages using different crawling approaches.

In Figure 7.2, the quality page retrieval rate from 3 different crawling methods are compared. The first method uses the quality index described in this chapter; the second method assumes that pages have the same quality score as their parent page, therefore only the un-weighted page-based scores from the parent page are used; and the third method uses vertical crawling with
Figure 7.2: The retrieval rate of high quality web pages using different crawling methods on a dataset consisting of 30,869 pages

no score at all. It appears that although the 3 lines show a significant amount of differences in the majority of the plot in Figure 7.2, they are very close at the beginning and at the end of the crawling process. The closeness in the beginning is due to the same set of seed pages supplied for the crawling task for ease of comparison, therefore the crawling processes would have to progress for some time before a difference in the crawling order can be observed. The gaps among the lines are much decreased towards the end as well; this is expected as the web pages in the dataset are eventually all retrieved, and therefore will be a common observation towards the end of most crawling experiments. The purpose for this focus crawling though, is to retrieve as many high quality pages, and as early as possible, because when applied to a much larger dataset or even to the web, the crawler will not be able to crawl exhaustively, but will only crawl a minute portion. This is the reason that the emphasis in consistently on the performance of the proposed strategy in the early stages of crawling. As is shown in Figure 7.2, the current method is able to retrieve a large portion of the quality web pages quite early in the crawling process, more so than the other methods, therefore, showing that the proposed approach is performing as expected, which is better than other approaches.

When the same experiment was conducted on a much larger dataset, the improvement in the quality page retrieval rate becomes evident. As illustrated in Figure 7.3, the retrieval rate of the proposed approach is at least twice as high as the other two approaches from the 500th entry to the 32000th entry, which is a significant improvement. It may be observed that although the dataset contains approximately 26.6 pages, only the early crawling progress is shown in Figure
The performances of the proposed crawler in both a small dataset and a much larger dataset both confirm that the focus crawler is able to retrieve highly scored high quality pages at a significantly higher rate, early in the crawling process than other approaches.

Survey result based weighting scheme

The same experiment is repeated using a new weighting scheme, which is based on the amount of agreement in the user survey. Details can be found in Chapter 6. Based on this weighting scheme, a final quality score can be obtained by multiplying the individual feature scores calculated on the child page, by their corresponding weight. The feature scores calculated on the actual pages are again summed up to produce a final score, but this time, instead of equal weighting, they use the weights from the amount of agreement in the user survey, as listed in Table 6.3. This final weighted quality score is used as the output of the MLP network and the elementary component scores calculated from the parent page without accessing the child page are used as the input for the MLP network. The setting which produced the best performance as discussed in Section 6.7 is one which incorporated a hidden layer consisting of 4 neurons, trained using standard back
propagation. The incorporation of the estimation weight will be slightly more challenging due to the fact that the network involved one hidden layer.

Again, this experiment aims to ensure that the quality score used to set crawling priorities are estimated as accurately to web pages’ actual evaluated quality score as possible. This is to verify that the focus crawler is correctly retrieving the web pages that match or exceed the quality threshold. The best performance of 0.00117 MSE was obtained during learning, and that appears promising; however, verification through the observation of the practical performance similar to that executed for the other weighting scheme will be required. This results in Figure ? which shows the rate at which web pages with an actual evaluated quality score of 70% or more are retrieved.

Note that the computed actual quality score may not necessarily refer to pages of high quality. What can be said is that the pages retrieved by the proposed method meet the quality criteria as were given by users. Whether the retrieved pages meet user expectations will be investigated in the next set of experiments.

7.5.2 Overall quality information retrieval system performance

The quality based intelligent crawler detailed up to this point has proven to successfully address the quality features as pointed out by survey participants and the literature, by accurately estimating the actual evaluated scores of target pages. However, the quality features stated as important may be important in theory, its importance in practice still needs verification. Since the actual score that all performances are evaluated against, is based on the feature scores and the a weight- ing scheme, both of which are derived from the survey result, there is a need to ensure that the actual score is indeed an appropriate measure of quality for web pages. It is quite possible that when asked, users are able to provide a set of important quality features, but then they do not use those features accordingly when actually evaluating the quality of a web page in practice. Therefore, a final step is required to verify the practical usefulness of such a system and its theoretically derived quality features.

Also, although the quality score estimation appears to perform well in theory, but as observed by [68, 142] incorporating some form of quality metrics generally improved the effectiveness of searching, therefore the performance also needs to be verified in a practical setting to observe the amount of improvement. This will provide an indication as to whether the performance improvement is significantly better than the general improvement observable with the incorporation of most metrics. Therefore, in order to ensure that the information retrieval system developed using the intelligent crawler embedded with quality features will be useful in a practical setting,
a testing platform needs to be developed. The testing platform will be for the searching and retrieval of high quality web documents, and will resemble a functional search system.

The testing platform is to allow the system to be tested by users and be compared against another information retrieval system, to observe whether there is an improvement and whether the improvement is significant. A web-based search comparison platform was then developed as a result, which uses a simple interface, allows users to perform standard searching using a query, and lists the top 10 results returned by our quality intelligent crawler and another search system of choice side-by-side. The other search system chosen for comparison is the open-source Nutch search system. Nutch is the largest and most popular of the few open source search systems currently available. The interface of the search comparison platform, developed for the this verification purpose is illustrated in Figure 7.4.

![Figure 7.4: A screenshot of the search comparison platform](image)

The procedure at the back-end of the search comparison platform is as follows: when users enter a query, the query is passed to the server side, so that searching within an index can be performed. Then the web pages matching the search query are sorted according to two ranking mechanisms of choice, depending on the purpose of the comparison. Only the top ten results from both ranking mechanisms are returned to the client’s side for display in the user’s browser. It should be noted that the search comparison platform was developed for experimental purposes. No attempt is made to compete with the size and scale of commercial search engines such as Google. Therefore, our search engine is not designed to exhibit the power and amount of feature as in commercial scale search systems commonly available on the current Web. For example, when more than 1 search query term is entered into the query field, the current system inserts the
boolean term “OR” between search terms as default. Also, phrase search or other boolean terms are currently not supported. Another issue to bear in mind is that the size of index in the search comparison platform is not comparable to the established search systems currently available on the web, as only approximately 26.62 million web pages from one of the testbed is indexed for this task.

The statistics of actions performed by the users, such as time of visit, query submitted, result entries clicked through, and the vote for the higher quality search system are logged. For the analysis, there were a number of phases:

1. Initial phase

2. Index expansion phase

3. Weight update phase

4. Comparison with pagerank phase

For the ranking process, the first 3 phases aim to compare the quality ranking to a relevance-based ranking scheme, which is that adopted by the open-source search engine called Nutch, through the utilization of Lucene library packages. To allow such a comparison, the top 100 relevant results that match the user query is firstly collected; Nutch orders the results based on relevance, and our quality search system orders the top 100 matching pages according to their calculated actual quality score. The calculation of the quality score for the ordering of web pages has been extensively discussions in this thesis in this chapter and in Chapter 6. The relevance score used by Nutch to order the search results is computed using Equation 7.1, and is based on the cosine distance or dot product between a document and the given query vector in a Vector Space Model (VSM).

The score is computed as follows, the query submitted \( q \) is treated as a vector containing 1 or more terms \( q = t_1, ..., t_n \) where \( t_1 \) to \( t_n \) refer to terms within the query \( q \), and \( d \) refers to a document. The first two segments of the equation \( TF(t_{ind}).IDF(t) \) is the commonly used Term Frequency - Inverse Document Frequency (TF-IDF) measure in information retrieval to identify the significant terms. The \( b(field(t)_{ind}) \) is the field boost set during indexing, which in the case for the experiments, is set to the default value of 1. The last segment in the equation \( n(field(t)_{ind}) \) refers to the normalized value of a field, given the number of terms within the field, which is computed during indexing. It can be seen from the equation that the Nutch score has a linear relationship with each of the components in the equation.
score(q, d) = \sum_{t \in q} TF(t_{ind}).IDF(t).b(field(t)_{ind}).n(field(t)_{ind}) \quad (7.1)

For the final phase of the search comparison web survey, the aim is to compare the quality ranking to a popularity-based ranking scheme, which is that adopted as the basis for ranking by Google. The equation for the calculation of pagerank was discussed in Chapter 2, therefore will not be repeated here. Individual discussions on each of the 4 phases in the search comparison web survey and the results obtained correspondingly will now follow.

**Phase 1. Initial phase**

The initial phase is limited to containing only approximately 12 million web pages in the index, this is due to the time consuming nature of the indexing process, therefore it is expected that a number of queries will have no matching result, which will lead to users to not vote after searching, as no comparison can take place in such a circumstance.

The finding confirms the suspicion with only 15 votes from a total of 51 submitted queries for the initial phase. This low voting rate of 0.29 is computed by dividing the number of queries submitted to the search system by the number of queries which resulted in a vote. A low voting rate such as that received from this initial phase means that there is a very small amount of data received from participants which can be used to evaluate the performance of the proposed ranking algorithm. From the votes received, only 40% of users considered the results provided by our quality ranking mechanism to be of higher quality, and the rest of users considered the relevance ranking by Nutch to be of higher quality. Since there are only two ranking approaches for comparison, a performance of exactly 50% indicates that the two ranking approaches received equal votes, and more than 50% indicates that the particular ranking approach out-performed the other.

An interesting trend was observed though, that the users’ judgement of the higher quality system is dependent on the length of the query. For example, when only considering the votes where one keyword is used in the query, a higher percentage of 55.56% of users considered the quality ranking results to be of higher quality than those provided by Nutch. This may be contributed to the focus on relevance by the Nutch, and since there is a limited number of indexed web pages, relevance of a webpage to the query term is crucial. The fact that our quality ranking scheme does not take relevance into consideration means that it is quite possible for users to receive results which are of high quality, but has limited relevance to their query, in which case, their information needs are not addressed. This could also be one of the reasons for Google’s popularity, as they often boast to have the largest index of web pages. The effect of their pagerank
ranking scheme may be maximized through their large index size.

**Phase 2. Index expansion phase**

The second phase is when an expanded index was provided to contain all the web pages in the testbed. As suspected earlier, the increased size of the index also increased the percentage of vote per query. The voting rate increased from 0.29 in the previous phase to 0.34 in this phase. The performance of our quality evaluation mechanism also seemed to perform better than in the previous phase. In the current index expansion phase, the percentage of vote for our quality evaluation mechanism increased very significantly to 63.64%, which is an increase of 23.64% from the initial phase. When using the increased index, our quality evaluation mechanism outperformed Nutch. This indicates the advantage and the importance of a ranking scheme other than relevance when the size of the index is large. This also confirms the need for an effective ranking scheme for the existing search systems, as the number of web pages indexed is larger than ever due to the size of the current web, therefore the effectiveness of the ranking scheme used to sort the large number of relevant web pages would determine the success of the search system.

The effect of the length of query is not so apparent anymore with an increased size of index. When only considering the query terms with a single keyword, the quality ranking received 66.67% of vote, which is two-third of votes. As it can be seen, there is no significant difference (only about 3% difference) between the percentage of votes received from searches with a single query term and those from a mix of query length. This implies that a similar level of performance can be achieved by the quality ranking approach regardless of the length of query.

Another observation from this phase is that some users voted without clicking into the result sets returned for their query. The validity of their vote is therefore questionable. The work by [105, 128] made a distinction between expected quality and evaluated quality, which seem to be applicable to this scenario. Expected quality refers to the quality based on predictive judgment, which in this case would be the voting carried out merely based on the limited information about the highly ranked matching web pages in the user interface, which includes the title, URL and the first few words of the corresponding web pages, without the analysis of the content. In contrast, evaluated quality refers to the quality based on evaluative judgment, which in this case are voting carried out after clicking into some of the highly ranked matching results, so that an evaluation of the content can take place to arrive at a decision about their quality. If the results from only the evaluated quality are considered, quality ranking received 71.43% of the 7 evaluated votes in total. It should be noted that although quality ranking performed well when only the votes
based on evaluated quality are considered, the number of votes is very limited to be considered a convincing result.

**Phase 3. Weight update phase**

The previous phases utilized the weighting scheme based on MLP; this phase utilized the weighting scheme based on the survey results. This third phase is entered to evaluate the difference in performance achievable by the 2 ranking schemes considered in this research project. This phase received a greatly increased amount of participation totalling 74 searches, in which the voting seem to be rather random. This observation of random voting resulted in a voting result close to 50% for both of the ranking approaches, where the quality rank received 49.25% votes overall.

Due to the increased participation, the analysis of the expected quality against the evaluated quality is possible. When only considering the voting based on expected quality where users did not analyze the content of the search result before making a decision to vote, quality ranking performed poorly, only receiving 40.43% of votes. However, when users voted after evaluating the actual content of the search result, therefore making an informed and better quality judgement, quality ranking received a significantly higher voting of 70.00% from the 20 evaluated votes. This performance is similar to that achieved in phase 2, and shows that the search results produced by the quality ranking approach can indeed out-perform a relevance based ranking approach when users evaluate the quality of search results.

The weighting schemes deployed by the quality ranking approach in phases 2 and 3 are able to be compared at this point. It may appear that using the weighting scheme derived from MLP performs better, however, considering the difference in the amount of participation in the 2 phases, the results obtained from phase 2 may not be very representative of the perception of search service users. Although phase 3 showed less favourable overall performance, it was seen that after the non-indicative votes from user’s expection of quality, which affected the results of phase 3 quite significantly, the quality ranking approach using the user survey as the basis for the weighting scheme is able to out-perform the relevance ranking by Nutch. As a result, the weighting scheme adopted in phase 3, which is derived from the amount of agreement in the user survey conducted by [75] will be the weighting scheme deployed for quality ranking in this research.

**Phase 4. Comparison with Pagerank**

After comparing the performance of the quality ranking against the relevance ranking by the open source search engine called Nutch, a comparison with a more comparable ranking approach
would provide more convincing evidence to the value of the quality ranking proposed in this research project. As a result, a comparison of quality ranking against Pagerank is carried out as phase 4 of the search comparison web survey.

Efforts were made to ensure the validity of the comparison. Pagerank scores were calculated on all the web pages in the index, and stored in a database, in the same manner which the quality score is obtained and stored. Since both ranking approaches do not take relevance into consideration, searches would first identify the matching web documents, then produce 2 identical result sets sorted in order of relevance. One of the result set would be reorganized according to their quality ranking, and the other according to their pagerank. The process for reorganizing the results should retain some of the relevance information, as a total re-sorting of the result sets according the ranking schemes is not desired, therefore in this case, a sliding window approach is used. In the sliding window approach, the results are reorganized according to their ranking order in groups, and the window size determines the size of this group. After a group has been reorganized, the window is shifted by a pre-determined position, so that a different set of web documents can be reorganized. This can be illustrated in Figure 7.5.

![Figure 7.5: An illustration of the sliding window concept](image)

The choice of window size and the amount of shift require some attention, as a window size that is too large may allow less relevant pages to appear at the top of the list, and a window size that is too small may not allow sufficient overlap between the windows. The amount of shift also has a similar effect. If the shift is too large, there will not be sufficient overlap between the windows, and a shift size that is too small may allow irrelevant web pages to appear at the top of the list. Sufficient overlap is important, as it ensures that lowly ranked web pages do not appear at the top of the list, even if it is highly relevant. Some preliminary investigation into the results produce by various window size and shift value indicated that a window size of 10 and a shift size of 3 as illustrated in Figure 7.5 is appropriate. Note that the sliding window approach adopted here also helps to significantly improve the sorting speed since it reduces the computational complexity of sorting the documents, and hence, is a scalable approach to the intended purpose of this approach.
The performance from this phase of the search comparison web survey revealed that users consider that search results sorted according to a combination of relevance and quality ordering is significantly better than the search results produced by sorting according to a combination of relevance and pagerank. This is indicated with 65.22% of votes for the quality and relevance combined ranking and 34.78% for pagerank and relevance combined ranking out of a total of 23 votes. Since both result sets used the same relevance ranking as the basis, it can be inferred that the decision for voting is based on the difference in the ranking produced by the quality ranking and pagerank. This is a significant finding as the quality ranking consistently out-performs other ranking schemes, from purely relevance-based ranking to popularity-based pagerank. This confirms that users find quality ranking more effective than other ranking schemes, and implies that the quality criteria which were obtained with a major influence from the user survey results are indeed correctly reflecting the criteria human users adopt in evaluating the quality of web pages. Furthermore, since the improvements obtained from quality ranking over both the relevance and popularity-based ranking schemes are significant and consistent, it can be said that the improvement in performance is considerably better than the general improvement observable from incorporating most matrices as pointed out by [68, 142]. The quality ranking is therefore shown to deliver search results which users believe to be of quality, higher than achievable by other automated ranking schemes.

It was seen that users voted for our quality ranking scheme in up to 70% of cases. But why did we not get a value closer to 100%? The reason is the limited size of our database spanning pages from only 8,623 domains. Many domains specialise on a certain topic (i.e. car rental), and hence, searches on a specific topic often resulted in a result set of pages from the same domain. Since pages within a domain are often written in a similar style, and hence, exhibit similarities in quality score, the consequence is that we are often limited in the ability to re-order pages by quality score. In other words, due to the relatively small size of our testbed, the result set created by our search engine is often very similar to the result set produces by Nutch. This in turn limits the user’s ability to make a decisive vote. In fact, a user is asked to vote even if both result sets are identical. The effect is a random voting component in the overall set of received votes. We observed that is roughly 50% of cases, the two result sets are either identical or very similar, and hence, the maximum level of performance reachable when using this dataset is \((50\% + 100\%)/2 = 75\%\). In other words, the given framework allows the achievement of a maximum of 75% performance. Our results have been very close to this maximum achievable performance level, and hence, it can be stated that:

1. The proposed approaches are very effective in reflecting human perceived quality of Web
documents.

2. The observed performances will increase with the size of a snapshot (since less and less random votes will be necessary).

7.6 Discussions and conclusions

The crawler implemented for the research project used a basic depth-first crawler as a basis, and developed an approach of merging a distributed crawler with a focus crawler. The result is a flexible, efficient and effective crawling application that has the advantages of both crawling approaches. The crawler also boasts some features that have not been explored previously, such as the incorporation of real-time score calculation and quality score estimation. The quality based intelligent crawler using the novel quality evaluation approach detailed in this paper is based on the findings of a targeted user survey on the perception of web page quality, and some also have support from the literature. The findings allow the identification of an actual score, which is a weighted sum of a number of quality features, where the weights are derived from the amount of agreement on the importance of individual features according to participating users. Through a 3 phase experimental procedure, it was made possible to

1. Identify quality features that are useful for making a score prediction.

2. Obtain weights for each of the quality feature to arrive at a estimated final score that is as close to the actual score as possible.

3. Observe the performance of the quality-based score prediction through the intelligent crawler.

The experimental results such as those shown in Figure 7.2 and Figure 7.3 demonstrated that the approach proposed in this research successfully addresses the quality criteria as pointed out by survey participants and the literature. The additional verification procedure of comparing survey participants’ theoretical quality perception to their practical quality judgment also indicates that the approach proposed in this research addresses the current need for a search system that provides quality assurance over the search results.

In most instances, the need for quality assurance increases with the size of the index, and as our experimental results show, the quality evaluation mechanism outperforms its counterpart (Nutch and Pagerank) when the number of index pages is relatively large. Also, the quality evaluation mechanism developed outperformed Nutch, when the performance comparison is restricted
to queries that only contain one term. It was pointed out in literature that a large portion of search engine users use only one term in the query when searching for information online [69, 37], therefore indicating that when applied to an existing practical information retrieval system on the web, the quality evaluation mechanism will perform well.

Also, we have indicated two possible weighting schemes in combining the quality features to arrive at a single quality score. In this thesis, we have used the equal weighting scheme based on MLP results, as well as the weighting scheme derived from a user survey, as revealed in our investigations. It was interesting to compare the results of these two weighting schemes, as they indicated a similar level of performance, both of which out-performed the ranking which is only based on relevance. The weighting scheme based on user survey results was decided to be adopted by the quality ranking system, from which a comparison with pagerank was performed. The comparison with pagerank also showed favourable results for quality ranking. Part of this chapter was published in the International Conference on Web Intelligence [70].

The tests carried out using the overall developed quality information retrieval system has shown to perform effectively in delivery search results that better match the user’s need for quality information. It also showed significant improvement in comparison to other existing systems such as the ranking schemes publicly known to be adopted by Nutch and Google. Therefore indicating the contribution of the developed automated quality ranking scheme.
Chapter 8

Related work

8.1 Introduction

During the design and implementation of a quality-focused information retrieval system, advances were also made by other research groups, with particular focus on the efficiency of data retrieval from the rapidly growing World Wide Web. As discussed in Chapter 2, information retrieval systems each used a different approach in the design of their crawling application and scoring algorithm, in order to maintain an effective search system and provide differentiation from other systems. The exact algorithms as are currently used in general information retrieval systems cannot be discussed in detail, as those information are often not disclosed for competitive reasons as well as to avoid misuse. Therefore, this chapter examines only the features and mechanisms from current information retrieval systems, from which information or documentation has recently become publicly available. The aim of this chapter is to discuss some of the current and emerging developments that are relevant to this research project. However, unlike the information contained in Chapter 2, these developments and advancements were carried out in parallel to this research project, therefore did not influence the design and development of the quality information retrieval prototype.

8.2 Changes in web search

As mentioned, the crawling applications and ranking algorithms adopted by various search engines are different in order to achieve differentiation from other search services, users may not always observe such a case. It is quite common that a feature in one search service is also adopted and accessible from a different search service. This is due to the merging and affiliating activities of search service providers, which have been an area of focus in recent years with purchases of
smaller scale search services and talks of merges by the more dominant search service providers. Therefore, to prevent comparing search services that are very similar because they are owned by the same company, it is necessary to first identify the representative search services which are uniquely owned. The amount of usage by the users will then be compared to identify the highly utilized services and to observe the changes that occurred in those services.

8.2.1 The provision of search services

The major search services identified in Chapter 2 include Google, Yahoo, Alta Vista, Ask Jeeves, Northern light and FAST, but a great deal of change has occurred during the period of this re-search, which resulted in a very different picture in the current provision of search services. The changes in the provision of search services have significant impacts on the users’ perspective of the web since search services provide the dominant means of discovering web pages. Web pages displayed highly in the result list are more likely to receive visits from users, and increase in popularity. As a result, search engines have strong influences on the specific portion of the web users access.

The current web search is shared among a smaller number of service providers than few years ago. The share of searches as of July 2006 is as indicated in 8.1. As it can be seen, Google is still the most popular search service, followed by Yahoo, MSN/Live search, AOL, ASK, then others in decreasing order [124]. It can be seen that there are 3 giants in the search scene, each attracting more than 10% of total searches. They are Google at 43.7%, Yahoo at 28.8% and Microsoft’s MSN/Live search at 12.8%.

One of the few things that did not change since 2003, is Google’s dominance of providing

![Figure 8.1: The share of searches among various search service providers in July 2006](image)
search services, but Google is no longer limited to providing search services, it currently also provides email, blog and online word processor. Google’s collaboration with other companies such as Amazon, enables unique features to allow searching within the full-content of books. Google’s take-over of a popular online video sharing tool - YouTube - allows it to explore and provide video search.

Yahoo has also undergone a significant change; its firm position as one of the most popular portal services allowed expansion of search in many of its services. It shifted its focus on web directory to include general web search. This was done effectively through the purchase of one of the previously dominant search service - Alta Vista. Yahoo also bought All the web and Overture. There were speculations of merging with Microsoft’s search service in order to compete with Google, later Google also became a possible merge partner; However, none of these merges took place.

Microsoft has successfully merged with MSN, which allowed the company originally based on designing operation systems to also provide web services. Microsoft merged its windows messenger with MSN messenger, and continued to use the portal, blog and its unpopular search services from MSN. The company took a step further and produced Live search which is significantly superior at allowing personalization than MSN search. Live search was also extended to provide search in more specific areas.

Merges and take-overs are not the privileges of these 3 major search service providers. AOL also merged with Netscape search, and Ask (previously known as ask Jeeves) are now under the same organization as Excite, iWon, My way and my web search [124]. The trend of web search services seems to indicate increasing number of search features, and joining of forces. The increasing number of search features could be developed with the aim to provide services to better match the various search purposes of users, or perhaps the general and basic web search facility has reached a glass ceiling where the indexing of a significant portion of the web is a constant challenge, and after years of searching for a mechanism for quality assurance, there is limited progress in developing a ranking algorithm which closely associates to the quality level of web pages. As pointed out by [8], the existing well-known search systems exhibit strong bias on the more accessible data, due to their adoption of link-based ranking algorithms. These are both possible reasons for the extensions made to allow more specific searches in addition to the basic web search.
8.2.2 Extensions on searching functionalities

As it was shown in the previous section, current information retrieval services are no longer limited to search of web pages. Continuous monitoring of some of the more widely utilized search services revealed that there is a trend of offering an increasing number of search services. For example, image, video, news, map and local search have now become a common feature in most major information retrieval services. These features allow specialized search, which are especially useful for conducting known-item searches.

There are other features which can only be found in specific information retrieval services. Table 8.1 contains a list of features against some of the major information retrieval services to show the availability of each feature in the information retrieval services. It is important to note that although Yahoo! does not have a feature to specifically search only within academic articles or blogs, those articles are searched through the general searching facility. Similarly, although Google does not have a feature specifically designed to search for local business, the advertising links on the right hand side of search results contain business within the specified locality, and are relevant to the search query. Question and answer features are also partially and indirectly addressed through its inclusion of Wikipedia documents in the general search.

Another observation to be made from Table 8.1 is that search services have unique features that cannot be found in others, such as the full text book search in Google and people search in Yahoo!. Google’s full text book search searches through the entire textual content within books, instead of the previous approach of limiting the search to only the referencing information and the general category of books. Yahoo’s people search utilizes an electronic version of white pages, which has information such as name, telephone number and address. Live search has a feature which allows a search through health and medical information, users could then refine the search according to the conditions, drugs or alternative medicine. There will also be a brief description about the health-related terminology and a list of web results matching the search query.

From Table 8.1, it appears that all three search services provide some form of popularity analysis, but they all address it from a different perspective. Google incorporates popularity analysis in its ranking of web pages through the Pagerank algorithm, which is used in its general search facility. Yahoo provides a web master tool through Alta Vista which allows the display of statistical information for a web page, such as the number of hyperlinks to the specified page, and the pages that contain those hyperlinks. This allows a clear report on how popular a web page is, in terms of its connectivity. Live Search also provides popularity analysis, but on people instead of web pages. The xRank feature ranks celebrities according to the number of searches about them. There is also a feature to show and compare the number of searches on celebrities.
over a period of time, so as to provide an indication of their popularity.

A final point worth mentioning is the translation features offered by most major information retrieval services. Translation feature allows searching across multiple languages, which essentially increases the amount of data available to search through for the information that matches the user query. Even though it has been shown that most users only refer to the top few results for a search query, search engines still consider a large amount of index to be important.

Table 8.1: Description of the unique features provided by some of the existing major information retrieval services

<table>
<thead>
<tr>
<th>Features \ search providers</th>
<th>Google</th>
<th>Yahoo!</th>
<th>Live search</th>
</tr>
</thead>
<tbody>
<tr>
<td>academic article search</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>blog search</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>local business search</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>question and answers</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>full text book search</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>health information search</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>people search</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>popularity analysis</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>translation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

It can be seen that the search purposes identified in the literature, namely background search, known-item search, decision task and many items task, can be addressed by one or more of these features. In fact, if a user is well aware of all the functionalities that search services currently provide, the user will be able to find a suitable tool for his/her search purpose. For example, product search can be used for online purchase or product comparison, map search can be utilized to locate an address, the basic search facility can be utilized for general information seeking, academic article search could be used to conduct research for a paper, video search could be used to find a video clip of interest for entertainment purpose, and the list goes on. This shows that search services have made significant effort to improve the searching experience for users. However, users do not make full use of the variety of search features available in practise. As pointed out by [107], the general search facility is still mostly used. Figure 8.2 shows the usage of the various search features provided by Google in December 2007.

Although these functionalities provide either more specific search or search over more number of web pages, it is important to note that the information used to determine whether the material, no matter if it is a video clip or a web document, matches the user query, is still based
on textual clues. Therefore, there is still the need to crawl the web and index the retrieved web pages together with other relevant data. This retrieval of information can only be carried out through crawling. The next section will discuss the improvements made to crawlers in the recent years.

### 8.3 Improvements in crawling efficiency

The amazing rate at which the World Wide Web is growing has been stressed throughout this thesis, not only is the size unimaginably large, there are also constant changes occurring in the web. At this point of the thesis, it should be common knowledge that search services do not actually allow searching to be carried out in real-time on the web, instead, search is carried out on a previously retrieved subset of the web. Combining these two points reveals why information retrieval from the web is such a challenging task to excel at. Crawlers deployed by search services have to continuously traverse the web in order to retrieve documents not yet stored in the repository or known documents that have been updated, but the size of the web prevents a significant portion of the web to be crawled in a timely manner. Since the retrieval cannot be carried out in a timely manner, some of the retrieved web documents could have already been out-of-date by the time they are indexed for searching.

The crawling throughput achievable as documented in the literature [64, 29, 42, 17] means
that a balance between the amount of data to crawl and the up-to-dateness (freshness) of crawled data needs to be maintained. Crawling a large amount of data requires time, therefore by the time the data can be re-crawled to check for updates, it is usually few weeks since the initially data were crawled. In fact, as pointed out by [95], the average refresh frequency of major search services is not encouraging. MSN search refreshes its indexed data at an average of 4 weeks, Google at 1 month interval in average, and AltaVista at 3 months interval in average. This does little to guarantee the degree of up-to-dateness of the search results. On the other hand, crawling a smaller portion of the web allows a round of crawling to be completed in a much shorter time, so that checks for updates can be carried out more frequently, which produces more up-to-date search results. The drawback is that the search is then carried out on a limited number of web pages. Such a compromise between the amount of data and the page freshness is made due to the bottleneck of crawling efficiency. As a result, some research efforts went into investigating approaches to speed up the crawling efficiency. A significant contribution in this research area is the publication by Lee et. al. [81] on IRLbot, described below.

The IRLbot developed by Lee et. al. [81] is considered a break-through in the crawling speed achievable by a crawler. IRLbot is a standalone crawler which utilizes the resources of a single powerful machine. It boasts the capability of being able to retrieve 6.6 billion web pages in 7 days, which equates to a throughput of approximately 10,913 pages per second. This is the highest throughput ever achieved by any crawler.

The IRLbot stores the list of URLs already seen on disk, as it is not possible to load the total amount of data (8 bytes hash × 6.6 billion URLs requires 44GB) into memory. The crawling priority for URLs are dependent on the pages already crawled and an allocated budget which is influenced by the number of inlinks to the page. IRLbot has crawling restrictions such as the following: does not follow redirection, does DNS caching, and allocates a budget of maximum number of pages permitted to crawl for a given Pay Level Domain (PLD). IRLbot, like other crawlers, also has few challenges that require addressing. The challenges will be described below, together with the solution proposed to address them.

- Complexity of verifying URL uniqueness

For each new URLs discovered, it is required to verify it against the list of URLs already crawled to decide whether to keep the URL or discard it. Since the number of pages IRLbot has to deal with is in billions, this task becomes very challenging and time consuming.

The solution proposed is to use a DRUM algorithm which takes the URL string and its 8-byte hash value, and checks the new URL against the list of URLs already seen (URLseen)
using bucket sort, then adds the new URL into “URLseen” and the “queue” of seeds if it
does not find a match in “URLseen”.

- The impact of multi-million-page sites

Some web sites contain so many web pages that they end up over-flowing the list of URLs
to crawl, and could even trap crawler indefinitely. These sites could be spam site for this
exact purpose, or it could be a legitimate large site.

The solution to address this issue is to use the STAR algorithm to calculate a reputation
score based on the number of inlinks per PLD. For example, http://www.google.com would
have google as its PLD, and http://www.uow.edu.au would have uow as its PLD. The rep-
utation scores set a limit to the maximum number of pages that is allowed to be retrieved
from a PLD, according to its inlink-based budget. The default budget is 10, and the top
10,000 PLDs receive a linearly interpolated budget between 10 and 10,000 pages.

- The processing of the crawling list

If the crawler has to periodically re-evaluate URL’s budget allocation, this would produce
little useful URL at the cost of huge overhead, and would result in the crawler making little
progress when the number of URLs to crawl becomes large.

The solution to address this issue is the BEAST algorithm, which enforces the budget
limit, and sets the crawling priority. This is done with the aim of distributing traffic, and
can be seen as a polite feature for web servers. The budget can be updated when reputation
changes as new inlinks are discovered during a crawl. PLDs with the total number of
known pages exceeding its budget will be crawled less frequently as the are more likely to
be spam, unless more inlinks are discovered to show that they are not.

Although IRLbot seems to be highly efficient, it cannot be utilized by this research project.
The reason for this is that the computing and network resources required for the execution of
IRLbot cannot be easily obtained. Also, IRLbot cannot be set-up in a distributed environment,
which is useful for distributing and localizing the network traffic. Although the paper proposed
that the efficiency of IRLbot in a distributed environment would be linearly increasing with the
number of participating machines, however that will not be the case. The communication among
the machines are required in order to avoid different machines crawling the same set of web
pages multiple times, and this communication overhead will be even more for IRLbot as it needs
to maintain a global view of the URLs already crawled, and keep track of the inlinks, reputation
and budget. These were not taken into account when the proposed distributed efficiency was
In addition, IRLbot’s measure of not being trapped in a crawling loop is through limiting the number of web pages per PLD. This, in contrast to the duplication detection mechanism in the crawler developed for this research project, does not prevent the multiple retrieval of duplicated information. As a result, the percentage of useful information retrieved will remain low for IRLbot, and the looping crawling trap engages the crawler at a slower pace, but is not prevented. These facts make IRLbot unsuitable for being adopted by this research project.

8.4 Web page ranking approaches

Current ranking approaches incorporated in existing search systems are usually semi-automated. They involve a combination of algorithms which allow automated calculation of scores, and some filtering process which require manual effort. The automatically calculated score is usually based on link analysis; quality-based ranking scheme is not yet incorporated into existing search systems. This section will analyze the well-documented pagerank algorithm, which is still being used by existing popular search service. A quality-based ranking scheme which can be automatically evaluated without human effort has been published recently. Although it has not been incorporated into existing search system, the automation of the approach provides it with the potential to be incorporated into a search system, therefore, this particular quality-based ranking scheme will also be investigated.

8.4.1 Link analysis based ranking approach

As indicated previously, Google is currently the most popular search service provider. The well-documented Pagerank algorithm is still being used by Google’s current search service, however Google also incorporates more than a large number of heuristics in addition to pagerank [14, 2]. Some manual involvement would also be necessary in order to identify the numerous attempts to manipulate its Pagerank algorithm [80, 2]. The capability for pagerank to address issues related to the quality of web pages is not yet known.

There is a reason that Pagerank is a successful ranking algorithm which has been utilized for many years. It is based on some reasonable assumptions, first of which is that when web masters create a hyperlink to another page, the target page must be of value to receive such a recognition by means of an inlink. When the number of hyperlinks in a web page increases, the value of each of the hyperlinks decreases, because the hyperlinks may no longer be a careful selection of an elite group of web pages that are considered important. This is implemented by sharing
the pagerank of a page among the target pages it links to, and the more hyperlinks there are, the smaller portion of the host pages’ pagerank that each target page will receive. The third assumption is that important web pages which are highly ranked should have more influence on what is of value. This is realized through the iterative process of calculating pagerank of target pages based on the host pages’ pagerank [14]. Although all these assumptions are reasonable, and it may appear that pagerank is a good indication of page quality to some [19], the resulting pagerank is in fact merely a rank of popularity. This will be discussed below through the identification of the shortcomings of pagerank.

The first shortcoming of Google’s pagerank is that it is vulnerable to rank manipulation. There has been many link spamming efforts to manipulate the pagerank of web pages in order to gain higher rankings than they should and have the web pages appear at the top of search results. There were also spamming attempts to ensure the spam web pages appear in as many search result as possible, or appear in the result of popular search queries, which may not be related to the content of the spam web pages. These are documented in [44, 127, 137].

The second shortcoming of Google’s pagerank is more related to this research project, as it clarifies the unsuitability of pagerank for quality evaluation. For the legitimate links that are not created for spamming purposes, the assignment of links is not always due to careful quality consideration. The hyperlinks may be created to refer to a web page of similar topic, a useful page for some resources, or a frequently accessing page such as a friend’s website or a forum of interest. This claim is supported by [125], in which it was pointed out that from an evaluation of web pages within university sites, the hyperlinks more often point to link-collection pages or subject specific resources, rather than high quality scientific material.

Due to the variety of reasons for creating hyperlinks, the initial assumption of pagelink which states that hyperlinks are created to point to web pages of value as a form of recognition may no longer hold true, therefore a discrepancy between web pages with high quality and those with high pagerank would be expected. An empirical comparison of pagerank and the quality score proposed as part of this research project on a testbed indeed showed limited correlation between the two. Due to the large number of web pages in the testbed (approximately 26.62 million), the ranking order of every thousandth web page are plotted in Figure 8.3. The data is sorted according to pagerank, therefore a perfect correlation would follow the diagonal line, and the points lower than the diagonal line would indicate that the pages are better in quality than indicated by pagerank, whereas points above the diagonal line indicate that the quality of the pages are not deserving of its high pagerank. However, Figure 8.3 shows a scatter of points away from the diagonal line, indicating that there is a very different ordering of the web pages by the
Figure 8.3: Illustration of the correlation between pagerank and the proposed quality score

two ranking schemes, and the two schemes may have a low correlation.

It may be observed from Figure 8.3 that some sort of diagonal pattern is occurring is small
segments, for example, between entry number 17 million and 26 million. This occurrence begin
at approximately the 10 million-th entry, and appears more obvious towards the end of the plot.
The reason for this occurrence is due to the fact that pagerank score can be calculated more ac-
curately on large datasets because of the more complete knowledge of the inlinks, but the dataset
used in this case is limited to 26.6 million web pages, which appears to provide insufficient in-
formation about the inlinks. This resulted in a large number of web pages appear to not have
any or have only a very small portion of inlinks, therefore correspondingly, received the default
pagerank score of 0.3 or a low pagerank score of close to the default value. In addition, the
scores are sorted before being plotted in Figure 8.3, and a large number of pages with all ranges
of quality score can be found with the low pagerank scores of close to or exactly 0.3, therefore
producing the effect of the diagonal line is small segments.

Another observation from the empirical analysis is that the highly ranked web pages accord-
ning to pagerank are mostly index pages or pages with a shallow directory, whereas the highly
ranked pages according to our quality score are pages usually deeper in the web site structure.
Table 8.2 illustrates this point by comparing the depth of the top highly ranked web pages, in a
testbed which consists of approximately 26.62 million pages. In the table, a depth of 1 refers to
the home page or index page of a domain, and the range of depth in the testbed is from 1 to 9. As
can be seen, the top highly ranked web pages according to quality score is consistently deeper
Table 8.2: The depth of top web pages as ranked by pagerank and the developed quality score

<table>
<thead>
<tr>
<th>Range</th>
<th>Depth according to pagerank</th>
<th>Depth according to quality score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Minimum</td>
</tr>
<tr>
<td>Top 20</td>
<td>1.75</td>
<td>1</td>
</tr>
<tr>
<td>Top 200</td>
<td>1.90</td>
<td>1</td>
</tr>
<tr>
<td>Top 2,000</td>
<td>2.43</td>
<td>1</td>
</tr>
<tr>
<td>Top 20,000</td>
<td>3.06</td>
<td>1</td>
</tr>
<tr>
<td>Top 200,000</td>
<td>4.95</td>
<td>1</td>
</tr>
</tbody>
</table>

The difference in depth is more noticeable when only the top few pages are selected. In Table 8.2 for example, when only considering the top 20 highly ranked web pages, the ranking by the developed quality score is approximately 2 levels deeper, and when considering the top 200 or the top 2000 highly ranked web pages, the difference in level for both of them is almost 4. The difference in the depth of web pages decreases when more web pages are incorporated into the analysis.

Also, in Table 8.2, the range of depth between the minimum and the maximum shows that there is no index page at all in the top 20 highly ranked web pages when the web pages in the testbed are ranked according to the quality score; however, further analysis showed that there are 4 index pages in the top 20 entries ranked according to pagerank. This finding is not unreasonable since informative web pages about a topic will not be expected at the index page level, however, those pages are expected to receive a large number of inlinks. This difference between high quality and highly popular web pages is also observed by [85], implying that pagerank which focuses on web pages with large numbers of inlinks cannot identify quality web pages, as it has no correspondence to quality, and therefore cannot be used for quality evaluation.

Not only has pagerank been shown to not correspond to page quality, it has also altered the web in an undesirable manner. The distribution of web pages according to their pagerank typically has a median much lower than the average, meaning that many pages have a low pagerank while only a minute portion of web pages have a high pagerank as illustrated in Figure 8.4. Those small portion of highly ranked web pages will usually be display at the top of search result lists, which receives high exposure, and since web users and web masters can be easily influenced by the preference of others, web pages with many inlinks are likely to receive further inlinks [10, 104, 31]. This results in the “rich gets richer” global pattern [86, 31].

From this discussion, it can be seen that although pagerank algorithm has been adopted by an existing search service for a long period of time, it has the shortcomings of being easily
Figure 8.4: The distribution of pagerank in the top 500,000 web pages of the 26.62 million pages testbed

manipulated, and unable to identify quality web pages. As a result, pagerank or any other link analysis alone, cannot evaluate the quality of web pages, because they only address popularity, and not quality.

8.4.2 Machine learning based quality evaluation

There have been a number of publications in recent times that propose methods of evaluating the quality of web pages. The difference between the quality evaluation approaches proposed in the past and in recent times is that researchers are increasingly aware of the magnitude of the web and the need for quality evaluation of web pages, therefore the recently proposed approaches no longer involve tasks that require extensive manual involvement, as it is widely accepted that the tasks requiring manual effort cannot be applied to a large scale of documents. Instead, recent approaches focus on measures that can be automatically calculated. One example is the approach proposed by [86], which supplies atomic measures that can be extracted from web pages into a machine learning algorithm for quality evaluation. The atomic measure include the following, categorized into 7 groups:

- File measures - length of URL, the length of HTML title and the file size
- Frequency of important HTML tags - such as H1, H2, H3, layer, table and others
- Measures based on HTML lists - number of lists, average, median and deviation of the number of \textit{li} tags per list
• Measures based on HTML tables - number of embedded tables, number of tables divided by file size average, median and deviation of the number of tr and td tags per table

• Colours - The number of colours, number of unique colours, RGB values of the most frequent colour, text colour and background colour

• Language features - number of words, number of unique words, number of stop words, number of sentence markers, and relation between words and stopwords

• Calculated measures and relations between atomic measures - number of out-links to file size, number of graphics to text length and others

There is a total of 113 atomic measures in the approach proposed by [86]. Not all measures are displayed here due to the large number, but it is interesting to observe that link analysis is not incorporated as one of the measures. The reason for the selection of these measures and the large number of measures is not documented. A machine learning algorithm is then trained on the measures extracted from web pages. Finally, a prototypical meta search engine is deployed to allow the evaluation of search effectiveness by users

There is very limited work in the area which proposes an automated quality evaluation approach suitable for web documents, therefore a comparison between the research conducted in this project and other related work on quality has not been carried out up to this point in the thesis. However, the work by [86] shows sufficient similarity with the approach proposed in this research project for a comparison.

Firstly, in both work, quality is consistently considered independent of relevance - the situational value of a web page for a specific information need. Secondly, both work agree that machine learning approach is the most promising approach to evaluate the quality of web pages in an automated manner, superior than compiling a strict set of rules on what constitutes high quality web pages. Lastly, both work developed a final system which is then assessed by users to judge its ability of performing quality-based information retrieval.

Although there is a few similarities between the quality evaluation approach proposed by Mandl and by this research project, however, there are differences between the two approaches as well. Firstly, the atomic measures proposed by Mandl [86] is unfounded, whereas the 12 quality criteria proposed in this research project is based on a set of user survey, and are the carefully selected quality criteria that users believe to be significant for the quality evaluation process. For example, it is unsure if the RGB value of the most frequent colour influences the quality of a web page at all, and if it does, how it influences the page quality. Also, how the author determines the amount by which the page quality changes as the RGB value differs is not clear. Therefore,
the lack of reasoning and support for the selection of quality evaluation measures results in the approach considered questionable. Secondly, the large number of measures incorporated into Mandl’s quality evaluation, although can be automatically extracted, is expected to be time-consuming. As a result, the application for Mandl’s quality evaluation approach on a significant scale of web data may not be feasible.

Although the quality evaluation approach proposed by Mandl [86] has some similarity of the approach proposed in this research; however, it will be challenging to be incorporated into a practical system for the purpose of quality evaluation, because there is no foundation to support the selection of measures, its effectiveness as compared to other search engines is not known, and the approach of calculating a large number of measures for each web page appears to be time-consuming for a large scale evaluation.

8.5 Conclusion

This chapter has investigated recent work on various relevant aspects of quality information retrieval. The number of unique search services has decreased, while the search functionalities provided increased. Advancement in crawling efficiency was made, and that is a significant contribution to the retrieval of data from the web, however, powerful and dedicated super-computer and high bandwidth would be required in order to achieve the throughput claimed in [81]. Recent work has proven that Google’s pagerank cannot indicate the quality of web pages, and work on quality information retrieval in the form of a ranking scheme based on quality evaluation has been carried out, but its contribution seems limited. It would seem that the work carried out by this research project would contribute to the research community, as it fills the gap between a theoretically founded quality evaluation approach and a practical information retrieval system, which even recent work failed to address.
Chapter 9

Discussion and conclusion

9.1 Introduction

This thesis provides evidence that there is a need for a quality-focused information retrieval system, through the observation of the characteristics of the World Wide Web, as well as the identification of a gap in the literature. The analysis of existing information retrieval systems confirmed the need for and the absence of an information retrieval system with the capability of evaluating the quality of web documents. As a result, a quality information retrieval system was proposed and developed, which showed very promising results.

The proposed quality information retrieval system, as described in this thesis, consists of a focus crawler with a quality score estimation mechanism. The system allows the crawled pages obtained in the early crawling process to contain a high number of target web pages above a quality threshold. This number of web pages above the quality threshold is significantly higher than those obtained through a standard breadth-first crawler. The significant performance improvement indicates a successful implementation of a focus crawler that is able to prioritise crawling in order to access the more promising web pages and retrieve target pages at a high rate. It also indicates that an effective quality score estimation mechanism for unseen pages based on properties of current pages, is able to correctly identify target pages before the web page is retrieved.

The quality evaluation mechanism is later utilized as a scoring and ranking component, to assist the ranking of web pages for a query. This revealed some interesting findings, such as the importance of a considerably large index, the importance of document relevance to the specified query in users’ judgment of document quality, the distinction between expected quality and evaluated quality, and the usefulness of a quality-based scoring system which reflect the decision process in human quality assessment.
9.2 Findings and implications

The development of a prototype for quality information retrieval on the World Wide Web as described in this thesis involved research processes which revealed interesting findings.

9.2.1 The need for automated quality information retrieval

The review of literature in Chapter 2 showed the processes required in the development of a search system, especially the back-end processes such as the crawling, indexing and ranking processes. Typical systems users are rarely aware of these back-end processes, but they affect the performance of the system significantly, and therefore require study and analysis to ensure an improvement in the performance and usefulness of search systems.

The literature review also revealed a gap between the theoretical quality models and the ranking schemes implementable in practical systems. This has resulted in the adoption of two different types of ranking schemes. The first is a manually-achieved quality ranking scheme which can only be applied to a limited number of web pages, and the changes to those pages could not be reflected in the system in a timely manner. Although this ranking scheme offers search results with higher quality, but it cannot be applicable to sufficient web pages for users to find it useful. The other ranking scheme is based on some algorithm which can automatically produce a score to assist in the ranking process. This type of algorithm is often based on the link structure among web pages such as Google’s pagerank. Although this ranking scheme is also to be applied to significantly larger number of web pages than the other ranking scheme, the quality of search results displayed to system users cannot be warranted.

These findings from the review of the literature imply that there is a requirement for search systems to incorporate an automated mechanism which assesses the level of quality in web pages. The quality criteria identified in the literature are based on the cognitive decision in the quality evaluation process performed by human, which need to, but could not be automated.

9.2.2 The requirements of information retrieval experiments

The testbed selection process during this research as discussed in Chapter 3 indicated the size and the dynamic nature of the web. As a result, some previously considered benchmark testbeds are no longer representative of the web. Some theories about the rate at which the web is changing are also no longer valid. New trends in the web such as the common use of a variety of document types, the composition of general TLDs, and the use and issues of scripts on web pages are identified.
The findings from the analysis of available testbeds in comparison to the characteristics of the web lead to the implication that a new set of testbeds is required from a practical crawler, which should be able to crawl efficiently, avoid the traps and problems the web may create for the crawling process, and most importantly, crawl effectively to provide an accurate representation of the web.

9.2.3 The requirements of the quality evaluation process

The analysis of documents through clustering and page examination produced several significant findings. Firstly, the features incorporated for the machine learning task is essential. If the features provided are indicative of the task, it could produce favourable results, otherwise, it may have an adverse effect. Not only is the selection of features important, the number of features incorporated for the task should also be carefully decided. This is because if insufficient features are selected, the best achievable performance may not be reached, but if the number of features is too large, it would prolong the training time unnecessarily. Therefore, it is vital to maintain a balance between the level of performance and time requirement.

The document clustering task also revealed that the amount of weight to associate with various components greatly influences the performance achievable. In the small-scale task carried out in Chapter 4, the components are the node labels and the structural information of the children nodes. However, in the quality evaluation task, the component would be the various quality criteria, therefore, the weight for the quality criteria would have to be carefully selected.

Another finding is that although the performance achieved for the clustering task by this research team is the best among a number of research teams, it only produced a Micro F1 of 0.38, and a Macro F1 of 0.34, which is a low accuracy. Further more, the tasks with content analysis in addition to the basic structure did not out-perform the structure-only approach, indicating that the content analysis approach can be much improved. It then follows that it would be beneficial to investigate deeper into the properties of web pages, as was carried out in Chapter 5. The finding from the web property investigation is that there are many features that can be extracted from web pages, however their significance to quality is unknown, and that some quality indicators cannot be extracted and analyzed in an automated manner.

These findings imply that properties of web pages should be carefully examined for their suitability to contribute to an algorithmic description of quality, as deriving an algorithmic description of a quality criteria is a challenging task. Unsupervised machine learning approach does not appear to be able to produce a satisfactory result, perhaps supervised machine learning approach would be more appropriate for this task. Also, a weighting scheme would be required
for each quality criteria, in order to form a final quality score which is reflected of the quality of a web page.

9.2.4 The overall quality information retrieval system

From the findings and implications mentioned previously, the requirements for the proposed prototype was be defined and refined in some cases. Based on the requirements, a prototypical system which could perform quality information retrieval was developed, and evaluated by users. The findings from the development show that an algorithmic interpretation of a cognitive concept of quality is extremely challenging. The approach taken appear to perform well, as the web survey on search comparison revealed in Chapter 7, but is by no means the only approach possible.

The testing and comparison of the overall developed system reveal that a considerately large index is required in order to allow the effect of ranking to be practical, and to provide the system with knowledge of a larger number of high quality web pages. Also, user perception of quality could be attributed to a user’s predictive judgement or the evaluative judgement. Furthermore, the overall system showed that the participants in the search comparison web survey consider the prototype developed for this research to be better than two of the existing search systems, namely Nutch and Google.

The findings from the development of the system imply that this research on quality information retrieval on the web has contributed to the body of literature in the topic, by providing a quality information retrieval system with a strong theoretical foundation. This developed system also has practical implications because it addresses the shortcomings of high-quality manual ranking approaches, and automated ranking approaches which does not offer quality assurance. The result is a search system which is able to evaluation web page quality in an automated manner, and produces a performance better than the most widely utilized search system - Google, as evaluated by web users who participated in the web survey. It should be noted however, that although the current developed system performs well, and is able to out-perform some existing information retrieval systems, it can still be further improved in the future.

9.3 Contributions

The research as presented through this thesis makes multiple contributions to the research area. As the web is evolving continuously and its properties are not fully documented, a number of areas were carefully examined during the development of a prototype for quality information
retrieval. Through these processes, the need for research was identified and many interesting findings were discovered. As a result, the following contributions were made as part of this research project.

Firstly of all, a scalable and accurate distributed crawler was designed and implemented for the World Wide Web. The crawler is designed to minimize network overhead, avoid approximations, and while at the same time, not requiring any specialized computing equipments. The developed crawler incorporates a duplication detection feature which is novel and not found in any other crawlers. Because of this feature, the crawler is able to retrieve information from the web in a more complete and accurate manner, therefore providing a different perspective of the web in the snapshots to that provided by other crawlers.

The second contribution is on the work carried out through the utilization of the crawler. The crawler enabled the creation of more up-to-date web snapshots, and in the process, the web properties, dynamics and trends were identified. This will provide awareness to the characteristics of the web, which will be useful for research in information retrieval area. The evolving nature of the web has resulted in a large degree of uncertainty in which theories are still valid, and which theories are out-of-date. The thesis contributed to the body of literature by exploring some of the theories, and investigated the degree to which they are still applicable to the current web. Other new dynamics and trends were also identified in this research.

The third contribution is a novel approach of estimating the quality of a web page before accessing the actual web page. As mentioned earlier, there seem to be a gap in literature between information retrieval and quality evaluation approaches. Our research contributes to closing this gap by addressing the topic area of quality-focused information retrieval, as well as proposing and developing one of the first information retrieval applications that has incorporated an automated quality evaluation mechanism based on the results of a user survey, into a focus crawler.

The final contribution of this research project is the developed quality evaluation algorithm, which can be applied as a scoring algorithm for ranking web pages. As was shown in Chapter 2.5, the scoring and ranking algorithms currently implemented in practical applications do not directly address and evaluate the quality of web pages, and they may be vulnerable to manipulation. However, the quality evaluation algorithm developed in this research is able to directly evaluate features in a web page, that have direct indication of its quality. Also, although the quality evaluation algorithm is also possible to fall victim of score manipulation, but in attempting to manipulate the score, the particular web pages would have to make modifications to the relevant feature such as increasing spelling correctness, improving content correctness and other quality criteria, which inevitably increase its quality. Therefore, the result of using a quality evaluation
algorithm as scoring or ranking algorithms is quite beneficial.

9.4 Limitations

During the research, there were constraints on time and the amount of resources available during the research. As a consequence, the research has several limitations.

- The largest testbed consists of approximately 26.62 million web pages, and not a larger portion of the World Wide Web

  This is due to the restricted bandwidth of the university network, the limited number of computers that are able to participate in crawling, and the financial cost of generating traffic for the data retrieval.

- The crawler was designed to perform with limited resources and unpredictable environment

  The reason for this is that considerations need to be taken into account about the time and resource constraints aforementioned, the storage and processing capacity of the crawling server, and the unstable characteristic of the university network where scheduled down time and power outage occurred frequently.

- The lack of comparison for the implementation method of the quality criteria

  This is because an algorithmic description of the criteria identified to be significant for quality evaluation is a novel approach in the area, which has not been attempted by previous work, therefore has no benchmark for comparison. It was recognized that the proposed approach is one of many ways of interpreting the quality criteria, which showed an improvement over other existing approaches. since there are limited number of attempts to quantify various quality criteria, a comparison of the various possible algorithmic interpretations was not conducted due to the challenge in carrying out the comparison, and the time required for the task.

9.5 Future work

This research is part of an ongoing project, as improvements and extensions can be made to address emerging issues or additional functionalities. Also, the World Wide Web is continuously increasing in size and changing, therefore the implementation would have to follow suit with existing information retrieval systems, so that it can continue to provide useful and high quality search results to users.
The extensions from this research may branch out in a number of directions. First of all, other machine learning techniques such as support vector machine could be explored in determining the mapping between the parent page quality features and the child page quality score. This would be an interesting aspect as to find out which machine learning approach performs better in terms of the estimation of the child page quality score. A more challenging approach would be to consider the more fundamental aspect of the proposed approach here: in this research, we have only built a model between the parent page features and the child page aggregate quality score. We have not explicitly made use of the topology of the connection between web pages, except that the child page is connected to the parent page. We have paid no attention to the wider issue of how the web pages are connected, and how such connections might influence the way in which quality web pages are found. There is an obvious way in which such a connection influences the discovery of quality web pages, through the simple mechanism which we have used in our intelligent crawler, in ranking the estimated quality scores of links (but without downloading the web pages yet). But the question is: if we are given a neighborhood of the connections of the web pages (here neighborhood is taken to mean the distance between web pages in terms of the number of intermediate links), could such knowledge lead to better and quicker discovery of quality web pages.

The second direction for future work is perhaps an even more challenging issue, which is to examine more critically the way in which we have used the survey results. Currently our ways of using the survey results represent one of many ways in which this set of survey reveals the user perception of what are quality features characterizing a web page or website. We could have used different ways to convert such qualitative descriptions and understandings into algorithmic implementations. These are interesting questions which would be fruitful areas of future research.

Another direction for future research is to investigate emerging applications which could adopt the quality information retrieval approach. There are areas other than search services that require the information retrieval process. One emerging service is the subscribe/publish service where users can subscribe to interested newsletters, usually provided by individual organizations, so that users can be notified of current information of interest, the latest product description or current promotions and specials from the particular organization. Work such as [143] explores the quality control of the subscribe/publish service to avoid the phenomenon of “information overload” on the users’ side. This can be considered yet another application for quality evaluation and filtering.
Bibliography


250


252


Appendix A

Report of an informal interview with a librarian

Report of an informal interview with a librarian at Edith Cowen University, WA, Australia

Date: 5/11/2004
Duration: 1.5 hours
Interviewed by: Shirlee-ann Knight and Milly Wei-Tsen Kc (Research partners in this research project)

1. Introduction of the project and how this interview will be used in the project.

The librarian agrees that the current search returns too many results which do not necessarily match user need. A discussion about why so may people return to Google even if the results do not always match the user need followed, with the conclusion that users believe that although the search service may not produce satisfactory results, it is the best method available to find information on the Internet. A point was also raised that there are many search engines on the Internet already; therefore the current project would have to have factors that differentiate itself from others. We explained that the differentiating factors would be a small number of query results, but all of which are of a high quality. The librarian comments that if the project can return results that match the information the user needs, it would be helpful and would reduce the frustration associated with searching.

2. Could you tell us a little about your background as a librarian?

There are 3 main types of librarians, the cataloguer, reference staff and user assistance staff. “Cataloguers are a different breed of their own the interviewee said, they are excellent in
working with databases. All aspects were explained during the library course/training, including cataloguing, working in database and the set of standards on subjects. The interviewing librarian has been teaching library courses and assisting students in the recent years.

3. **How do you think librarians searching approach differs from the average user?**

The mindset is different. Librarians think in Boolean and use Boolean a lot during searching. Librarians also consider which categories the queried information would fall under and which source would be most appropriate to conduct the search in. The keywords used for searching is also different. A specialist librarian would know what keywords (crucial ones) to use during searching; a general librarian would need to know the topic well, or use a thesaurus to search effectively. The librarian recommends the advanced search fields for the less experienced searcher, as it is actually more guided and gives better results.

4. **What are the search rules or features in search engines and databases that you consider useful for searching?**

   - Allowing the search of query through the index and content of the database
   - The recognition and differentiation between keywords and phrases.
   - Two words without AND is treated as a phrase, and more than 2 words will have AND inserted in between automatically, and each word is treated as a keyword. Therefore, even if a person enters a sentence, each word is used as an individual keyword during searching
   - Some database supports spelling correction or searching of related words, these are useful as well.

5. **Can you describe the current library system?**

The current classification system used is the Universal Decimal Classification (UDC), however, other specialist areas such as law or medical have variations. The organization of subjects and headings follows the Library of Congress standard, where the subject category and author are controlled (follow the format and classification used in the existing system), and others are not controlled, but follow rules for certain conditions (for example, punctuation and abbreviation).

6. **What are the changes or trends in library systems?**
All cataloging used to be done by the librarians in ECU, but now, the library has to use copy cataloging (adopt records that has been brought down), due to the movement of outsourcing (buying records) using specialist group for some subject areas, to improve productivity. Future direction of library system would have 24 hours e-reference collaboration project to allow online librarian assistance any time of the day. This will be effective when the collaboration is done with a university of a very different time-zone.

7. **Are there issues with the current library system?**

The current outsourcing strategy presents several problems, as the libraries have to continue using the format used by external provider, which may create different references to the same book. In an electronic environment, it is possible to have multiple headings or numbers associated with a book, but in the physical library environment, it is not possible. In addition, modification can no longer be maintained by the general librarian, but will require specialists.

8. **What would be the factors determining the introduction of a new book or journal?**

For books, as long as there is a user request, it will be looked into. If the requested book is too specialised, an approval from an authority (for example, the research supervisor) would be needed. As for journals, the dataset group would propose a database, which is sent to the library, then the manager of purchasing area checks licenses and whether the technology fits (e-books should be able to work in the Innopac system, which is an integrated system used for cataloguing, circulation and purchasing).

9. **What are the processes involved in categorizing new books?**

The processing of a new book is done at the central processing area following the procedure below

(a) Look at citation information on the title page and the abstract. If the author has previous work, the format of the old catalogue should be followed.

(b) Then look at the content and check for keywords

(c) The book is then allocated at a place that can be easily obtained, with related books (keeping in mind the type of people who will access it)

10. **As can be observed in the World Wide Web, new terminologies and topics can be introduced and used widely in a short period of time. What would be the condition of introducing a new category in the library system?**
Library of Congress has its own anthology. ECU follows Library of Congress’s national standard, where changes occur globally.

From this interview, there are a number of ideas which may be utilized in our research project. For example,

- Specify the category at query stage
- Give option to specify keyword or phrase searching or carry it out as suggested by the interviewee
- Find appropriate source/community for different categories
- Need to consider people who know exactly what they want
- Need to be able to recognise phrase (may need to check for word relations and history of some terminologies or abbreviations)