A TEACHABLE SEMI-AUTOMATIC WEB INFORMATION EXTRACTION SYSTEM BASED ON EVOLVED REGULAR EXPRESSION PATTERNS

by

Nor Zainah Siau

A Doctoral Thesis

Submitted in partial fulfilment
of the requirements for the award of

Doctor of Philosophy
of
Loughborough University

October 2013

Acknowledgement

In the name of Allah, the Most Gracious and the Most Merciful.
Alhamdulillah, all praises to Allah for the strengths and His blessing in completing this thesis.

Firstly, I would like to express my gratitude to the Government of Brunei Darussalam for funding my PhD studies and looking after my welfare while I was in the UK.

I was blessed to have been supervised by Professor Christopher Hinde and Dr. Roger Stone for providing me an opportunity to undertake and trust me with this research and for their continuous support and encouragement. Most importantly, for steering me to a better direction even when I was not so sure where I was heading. My sincere thanks to Professor Ray Dawson (my internal reviewer), who showed me how I could improve my thesis.

I would like to thank my beloved husband, Sharul Tazrajiman Haji Tajuddin, who was always there for me, not only during my happy times but also my worst times, especially when he is also studying his own PhD. Also without whom, I never would have started my PhD and I never could have finished, either. To my beautiful princesses; Nurul Sa'adatul Nazirah, Nurul Farzana and Nurul Imanina for the love and endless support whilst I have worked on this thesis, even though they didn't understand my struggles. They made me forget my stress and inspired me to keep on moving.

My special thanks also go to my parents and my parents-in-law in Brunei Darussalam for all the prayers, all the inspiring words and moral support. Not forgetting my elder sister, Hjh Asimah for all the help and the rest of the family for keeping in touch.

My heartfelt appreciation goes to all my friends, who were with me during this journey, for the friendships and all the support. And finally, the Department of Computer Science, Loughborough University, for supporting my conference expenses and the resources.

Abstract

This thesis explores Web Information Extraction (WIE) and how it has been used in decision making and to support businesses in their daily operations. The research focuses on a WIE system based on Genetic Programming (GP) with an extensible model to enhance the automatic extractor. This uses a human as a teacher to identify and extract relevant information from the semi-structured HTML webpages. As web pages are dynamic and their structures differ it is difficult, if not impossible, to design and implement a generic automatic solution. This is especially true for reasoning and sifting through large amounts of data. Therefore, to achieve this enhancement, a robust information extraction system has to be taught rules and the relevant knowledge to understand different presentations of information.

The combination of textual and structural patterns is one of the many successful methods in WIE, especially with semi-structured data, without the need to eliminate noisy data. The key feature of this study is the use of semi-supervised learning to train the WIE system by providing training data. Regular expressions, which have been chosen as the pattern matching tool, are automatically generated based on the training data to provide an improved grammar and lexicon. This particularly benefits the GP system which may need to extend its lexicon in the presence of new tokens in the web pages. These tokens allow the GP method to produce new extraction patterns for new requirements.

The thesis describes experiments testing the GP method, introducing the concept of ‘clean grammar’ to avoid unnecessary overheads for fixing the evolved solution. It also assesses the semi-automatic approach using online training provider webpages with the task of extracting course title, date, location and price. To evaluate the effectiveness of this approach, a prototype system called Teachable Semi-automatic Web Information Extraction (TS-WIE) is produced. This system is used for pattern generation based on lexical and structural analysis. There are four significant challenges encountered to

improve the quality of the extracted information and to scale up the extraction. These are dealing with human assistance, managing the different structures of web pages, coping with the evolution of Web technologies, and developing a more efficient GP method to build new extraction patterns.

All four challenges have been overcome and the results show that the TS-WIE extractor achieved a precision above 95% for both extraction of the course title and location and an F-Measure of 99% and 80% respectively. The precision of the date and price is 100% and over 79% and 91% respectively for the F-Measure. The incremented grammar as a result of TS-WIE system shows that 75% of the websites tested have significant improvements ranging from 63% to 100% hits for all the course attributes.

Keywords: TS-WIE, dynamic grammar definition, Genetic Programming, regular expressions pattern and structural pattern (DOM).

Publications

Conference Publications

Siau, N.Z., Hinde, C.J. and **Stone, R.G.**, "An evolution of a complete program using XML-based Grammar Definition", *Proceedings of the 4th International Conference on Evolutionary Computation Theory and Applications*, Barcelona, Spain, Oct. 2012.

Siau, N.Z., Hinde, C.J. and Stone, R.G., "A Stepwise Evolution of Functions", *Proceedings of the 7th WSEAS International Conference on COMPUTER ENGINEERING and APPLICATIONS (CEA13),* Milan, Italy, January 9-11, 2013.

Table of Contents

Acknowledgement i
Abstract ii
Publications iv
Table of Contents v
Abbreviations viii
List of Tables x
List of Figures xii

Chapter 1
Introduction 1
1.1 Chapter Overview 1
1.2 Research Background and Motivation 2
1.3 Research Scope 5
1.4 Subject Domain – Case Study 6
1.5 Research Aim, Research Questions and Hypothesis 7
1.6 Significance of the study 10
1.7 Thesis Layout 11
1.8 Chapter Summary 14

Chapter 2
Review of the Literature 15
2.1 Chapter Overview 15
2.2 Web Information Extraction (WIE) and its building blocks 16
2.3 WIE to support organisations' events 20
2.4 Semi-automatic WIE 22
2.4.1 Introduction 22
2.4.2 Wrapper 25
2.4.3 Extraction Techniques 30
2.4.4 Evaluation of Information Extraction Systems 39
2.4.5 Challenges of WIE 41
2.5 Web Information Extraction Methods 42
2.5.1 Introduction 42
2.5.2 Extraction using Machine Learning 42
2.5.3 Extraction using NLP 44
2.5.4 Extraction using Ontology 45
2.5.5 Extraction using Genetic Programming 46
2.6 Evolving Computer Programs 47
2.6.1 Introduction 47
2.6.2 Genetic Programming Terminology 49
2.6.3 Basic Concepts of Genetic Programming 49
2.7 Variation of Genetic Programming 56
2.8 Critique 59
2.9 Chapter Summary 62

Chapter 3
Use of Genetic Programming to Evolve Patterns 64
3.1 Chapter Overview 64
3.2 The Evolution of Complete Software Systems 64
3.2.1 Introduction 64
3.2.2 “Sorting program” evolution - The work of Withall 65
3.2.3 “Sorting program” evolution - The work of Xhemali 68
3.2.4 Discussion 69
3.2.5 Critique 70
3.3 Automatic WIR/WIE System - Xhemali 72
3.3.1 Introduction 72
3.3.2 Automatic WIR module 73
3.3.3 Automatic WIE module 73
3.3.4 Discussion 82
3.3.5 Critique 83
3.4 Chapter Summary 84

Chapter 4
Practical Application of GP – Domain 1 86
4.1 Chapter Overview 86
4.2 Context 87
4.3 Evolutionary System Approach 88
4.3.1 Introduction 88
4.3.2 Grammar Validation Tool 89
4.3.3 Genotype to Phenotype Mapping Method 91
4.3.4 Programming Language 95
4.3.5 Test Environment 97
4.4 A Computer Program Evolution (CoPE) 100
4.4.1 Introduction 100
4.4.2 Sorting Lists of Integers 101
4.4.3 Sorting Lists of Integers (Seeded) 115
4.4.4 Reverse-Sort Lists of Integers (Seeded) 117
4.4.5 Distance from Mean 121
4.4.6 Results Summary 124
4.5 Chapter Summary 125

Chapter 5
Practical Application of GP – Domain 2 126
5.1 Chapter Overview 126
5.2 A Regular Expression Evolution (REGEXEV) 127
5.2.1 Introduction 127
5.2.2 Web page structure of the Training courses website 127
5.2.3 Regular Expression to define web data pattern 131
5.2.4 Regular Expression based on Extraction Rules 142
5.3 Related Work 153
5.4 Chapter Summary 155

Chapter 6
Practical Application of Teachable Semi-Automatic WIE 157
6.1 Chapter Overview 157
6.2 Target Schema 158
6.3 Methodology Adopted 159
6.4 TS-WIE System Overview 163
6.5 TS-WIE System Design and Implementation 165
6.5.1 Introduction 165
6.5.2 Graphical User Interface 168
6.5.3 System Design 172
6.6 Experiments and Results Discussion 185
6.7 Challenges 194
6.8 REGEXEV experiment revisited 197
6.9 Chapter Summary 200

Chapter 7
Conclusions 202
7.1 Chapter Overview 202
7.2 Summary of the Key Contributions 202
7.3 Limitation 206
7.4 Further work 208

References 210
Appendix 1 232
Appendix 2 239
Appendix 3 242
Appendix 4 247
Appendix 5 250
Appendix 6 253
Appendix 7 258

Abbreviations

AJAX	-	Asynchronous JavaScript and XML
API	-	Application Programming Interface
ATM	-	Apricot Training Management
BNF	-	Backus Normal Form or Backus–Naur Form
CoPE	-	Computer Programs Evolution
CRM	-	Customer Relationship Management
DARPA	-	Defense Advanced Research Projects Agency
DOM	-	Document Object Model
DNS	-	Domain Name System
DTD	-	Document Type Definition
EC	-	Evolutionary Computation
GP	-	Genetic Programming
GUI	-	Graphical User Interface
HTML	-	HyperText Mark-up Language
IDE	-	Integrated Development Environment
IE	-	Information Extraction
IS	-	Information System
IT	-	Information Technology
JSON	-	JavaScript Object Notation
ML	-	Machine Learning
MUC	-	Message Understanding Conference
NLP	-	Natural Language Processing
OBIE	-	Ontology-Based Information Extraction
PAT tree	-	Patricia tree or suffix tree
PDF	-	Adobe Portable Document Format file
PHP	-	PHP: Hypertext Pre-processor
REGEXEV	-	Regular Expressions Evolution
TS-WIE	-	Teachable Semi-automatic Web Information Extraction system

URL	-	Uniform Resource Locator
W3C	-	World Wide Web Consortium
WIE	-	Web Information Extraction
WIR	-	Web Information Retrieval
XLS	-	Microsoft Excel file format (spreadsheet file)
XML	-	eXtensible Mark-up Language

List of Tables

1	The objectives in relation to the methods and research questions …….…	9
2.1	Sample HTML structure to display the data…….…..…	31
2.2	Basic regular expression notation…….…..……	38
2.3	Confusion Matrix…….…….	40
2.4	Meaning of Evolutionary Computation terms for this thesis...…..……	49
2.5	Example of parents selected from parent population….……	54
2.6	Application of a Uniform crossover operator to the offspring……..	54
2.7	Application of a mutation operator after the crossover……..	54
3.1	GP methods - Withall versus Xhemali……...……	70
4.1	The characteristics of the experiments carried out in this research…...….	99
4.2	Comparison of results with the previous works (10 runs) – E1: Withall et. al. (2003), E2: Xhemali et. al.(2010b), E3: A replication of E1 with clean grammar, E4: A replication of E3 with multi-objective fitness function……..	112
4.3	Comparison of 100 seeds results between Withall's works (E1), clean grammar (E3), multi-objective fitness function (E4) based on generation cycle and time to achieve fit solutions…….	114
4.4	Summary statistics of the generations required to find a solution (100 runs) for Random, Solution Seeded and Solution Mutated……	116
4.5	A comparison between the reverse-sort program and the optimised reverse-sort program using evolved functions (measured in terms of the number of required generation)…….	119
4.6	The results of the DistanceFromMean……...……..	124
5.1	The number of web pages showing how the specific data is constructed..	130
5.2	Price representation on web pages……..……	130
5.3	Title of Course representation on web pages…….…….……	130
5.4	Regular Expressions used to define the patterns of the specific piece of data……..	132
5.5	A Summary of fitness criterion applicable to the extraction attributes.….	134
5.6	Sample websites to test the algorithm……..……...	144
5.7	Successfully evolved regular expressions to match the data of a course....	148

5.8 (a-d). Result of matching regular expression to the course attributes based on the % of hits, best generations, average generations and median generation applied to each URL. Note that '-' indicates that the attribute is not available on the web page. The 'managementtrainingcoursesuk. co.uk' attributes (excluding title) comes from a linked webpage; external to the current webpage of interest. 'ontargetlearning.co.uk' and 'medicalinterviewsuk.co.uk' locations are labelled using some specific names of particular places other than the city name e.g. hotel name and room name, which failed to be recognised by the system……..… 149

6.1 List of URLs relevant to test and evaluate the TS-WIE System to improve the quality of the extracted information..…….. 158

6.2 The research methodology for the TS-WIE system……..... 159

6.3 Link_status value, which is used to indicate the status of the relevant web pages been retrieved. (Source Xhemali 2010a)……... 172

6.4 Example of jQuery conversion to regular expression……...…… 180

6.5 Example of text conversion to regular expression…….…… 180

6.6 Results of experiment in % for Title extraction……...…….…. 187

6.7 Results of experiment in % for Date extraction……..…….….. 188

6.8 Results of experiment in % for Location extraction……..……..... 190

6.9 Results of experiment in % for Price extraction……. 191

6.10 Average performance in % of the TS-WIE system …….……... 192

6.11 (a-d) A repeat of REGEX experiment to URLs in Table 4.16 that have precision of less that 100%. The generation of regular expressions are based on the incremented extraction rules by TS-WIE system ……. 197

List of Figures

1	Thesis Structure ……..…….……	13
2.1	Performance trade-off relative to specificity and complexity…….….....	19
2.2	General view of Wrapper Induction System illustrating the four categories of system's degree of automation; manual, unsupervised, supervised and semi-supervised..……..……	26
2.3	DOM representation of HTML tags…….	32
2.4	HTML DOM node tree and relationship between nodes…..……...	33
2.5	Example of HTML structure ……..……	34
2.6	Example of jQuery path …….	35
2.7	(A) The basic GP flow diagram (B) The basic GP Algorithm ……..	50
3.1	An extract of rules stored in a PERL file to assist the generation of 'sorting' program…….…….……	66
3.2	An illustration of the formation of a phenome from a genome …...……	67
3.3	Rules stored in a XML file to assist the generation of a 'sorting' program ……...……	68
3.4	Xhemali's Automatic WIR/WIE System overview ……..…….	72
3.5	The XML-based Grammar rules (Xhemali et al. 2010b)…….……	77
3.6	The database structure used by the automatic WIR/WIE system ……...	79
4.1	Experimental Route showing the progression of the tasks in this research and the relationship between the components…….……..	88
4.2	A DTD that defines the structure of XML-based rules with legal building blocks (elements and attributes). The DTD is declared within the XML document as an internal subset and it ensures compatibility with the XML systems……..…...	90
4.3	Index of elements in 'if statement' rule of type 'sequence'…….	93
4.4	Index of elements in 'statement' rule of type 'selection' ……	94
4.5	Time vs memory consumptions by the evolutionary system in PHP…..	96
4.6	Formal specification of sort (source Cooke, J. 2004)……..	103
4.7	Simplified fitness function for sort (source Withall 2003)……..	104
4.8	Grammar rules are expressed in BNF form. In the actual implementation, these grammar rules are presented in XML format…..	106
4.9	An extract of XML-based grammar coded in PERL to guide the transformation of genotype to phenotype…….	107
4.10	Active genes versus padding……	109

4.11	Pseudo-code of the Genetic Programming to evolve a program……..	110
4.12	Genome to Phenome Mapping with a new grammar rule (block statements). This helps the mapper to decide if a particular statement (if, for or doublefor) has a single true statement or multiple true statements…….	111
4.13	A sample of an evolved Sort program generated by the evolutionary program……..…….	113
4.14	The extended version of the program statements syntax expressed in BNF…….....	119
4.15	PERL: An Example solution for a *swap* function embedded in a Reverse-sort program generated by the evolutionary program……..….	120
4.16	Specification for *DistanceFromMean* problem……...	121
4.17	Fitness Function for *DistanceFromMean* problem……..	121
4.18	Program statements syntax expressed in BNF for the DistanceFromMean problem……	122
4.19	An example of successfully evolved DistanceFromMean program incorporating a sequence of two loops ……..……..	123
5.1	(a-d) Sample information presentation styles……..	128
5.2	General structure of regular expression pattern..…….	140
5.3	Regular Expression Grammar rules are expressed in BNF form. In the actual implementation, these grammar rules are presented in XML format……..…….	141
5.4	An extract of *Grammar definitions* to evolve regular expressions …..…	143
6.1	System Developing Life Cycle with Prototyping (source Carey 1990)...	160
6.2	TS-WIE system supporting the automatic WIE system……...	162
6.3	A screenshot of the user interface for selecting the URL to process…...	167
6.4	Screenshot of http://www.capita-ld.co.uk/courses/Pages/absence-management-training-courses.aspx web page.....	168
6.5	A screenshot of the user interface for making selection and extraction of the relevant course attributes…….……..	169
6.6	TS-WIE system in relation to flow of control between processes……...	171
6.7	Accept Training Examples & Extract process of the TS-WIE System ...	178
6.8	Data Model Components of TS-WIE System……..	183
6.9	The system's response by highlighting all attribute values in pink that matches the provided examples (in a multi-instance attributes web page environment)……	185
6.10	The system's output after the task is completed. The data is stored first in the database before it is displayed back on the screen…….……	185

Chapter 1

Introduction

1.1 Chapter Overview

There is a growing concern over the quality of the extracted information from web sources and the adaptability of the extraction systems to various domains (Seidler & Schil 2011) including the resilience to cope with changes in the web page (Laender et al. 2002). Several innovations have been put forward to tackle these issues, ranging from semi-automatic with human intervention, to fully automatic solutions (Chang et al. 2006).

The World Economic Forum's Global Agenda Council on Emerging Technologies (GAC 2012) listed 'Informatics for adding value to information' as the top in the list of 10 emerging technologies for 2012. This item emphasised the issue of information overloading and the information extraction process as one of the main methods to provide good quality information. The full statement is stated as:

> 'The quantity of information now available to individuals and organizations is unprecedented in human history, and the rate of information generation continues to grow exponentially. Yet, the sheer volume of information is in danger of creating more noise than value, and as a result limiting its effective use. Innovations in how information is organized, mined and processed hold the key to filtering out the noise and using the growing wealth of global information to address emerging challenges.'

Numerous innovations for information extraction have been proposed and developed (Eikvil 1999, Sarawagi 2008), such as STALKER (Muslea et al. 1999), IEPAD (Chang et al. 2003), KnowItAll (Etzioni et al. 2005), TEXTRUNNER (Banko et al. 2007) and ReLIE (Li et al. 2008). Methods are

proposed to address various challenges ranging from a simple noise filtration (Meng et al. 2003) to improving the imprecision (Ipeirotis & Jain 2008) to managing uncertainty (Michelakis et al. 2009). The use of a Regular Expression Wrapper is one of the dominant techniques used in IE. Manually crafting regular expressions for WIE, is an error prone and expert-dependent task (Li et al. 2008). However, there have only been a few works done to evolve the regular expressions used in this area, such as Cetinkaya (2007), Barrero et al. (2009) and Xhemali (2010a).

This thesis addresses the topic of Web Information Extraction by manipulating the extraction patterns with particular emphasis on the Genetic Programming (GP) system used to generate the regular expressions used to extract the information from the web. The GP may need to extend its lexicon in the presence of new tokens in the web pages. No research of this nature has been found in the literature. This chapter introduces the context of the research with a brief overview of the research background, including the background information and the issues related to the justification of this research. The research aim, objectives and the research questions follow, which leads to the research hypothesis and the significance of the study. The structure of this thesis is outlined at the end of this chapter.

1.2 Research Background and Motivation

The World Wide Web (Web) has become the de-facto dynamic repository of information and has emerged as an important source of information. A report by Internet Systems Consortium (ISC 2012) shows that the number of hosts on the Internet has reached slightly over 900 million. There are 131 billion online searches per month worldwide (Comscore 2010) and 61% of the users worldwide do product research (IPSOS 2013). The rate of information generated on the Web continues to grow exponentially and is in danger of containing ‘more noise than value’ (Martin 2005), and it may become a serious issue for organisations worldwide to find it, make sense of it, organise it, and filter it (Allen & Wilson, 2003). A survey by WorldOne Research (2008)

reports that 62% of professionals in America have spent a large percentage of time searching information on the web and moreover, according to Feldman (2004), International Data Corporation estimates that knowledge workers spend 15–35% of time searching for information, yet are only successful less than 50% of the time. Substantial time is wasted due to the reliance upon search tools that could not provide sufficient precision (Hammer et al. 1997b), thus resulting in a huge waste of organisation's resources (Feldman 2004). This shows that information extraction is an essential tool, not only for individuals but also organisations.

Generally, Information Extraction (IE) is a technology that allows for the extraction of pieces of relevant information required by the user (NIST 2005). Information is said to be relevant if it meets the guidelines as to what kind of information the system should capture and this is normally domain specific (Eikvil 1999). Originally, traditional IE is based on text analysis using Natural Language Processing (Cowie & Lehnert 1996) to extract named entities such as person names and addresses from various sources within millions of potentially relevant documents. Modern IE technology now has shifted the focus to a wide variety of multimedia content such as images, audio and video and non-English language sources.

Web Information Extraction (WIE) is another form of IE but the extraction is strictly from sources on the Web. Unlike traditional IE, WIE relies on HTML mark-up tags and other delimiters (Grishman & Sundheim 1996), visual appearance (Cohen et al. 2003b) and involves huge scale of web sources and domains (Yates 2007). Given the complex and diverse nature of information presentation, achieving high accuracy and domain independency is difficult. Most WIE solutions are developed for specific domains (Banko et al. 2009), with defined sets of rules or data templates; therefore, a small change to a web page can cause imprecise results.

WIE, in recent years, has become a challenging field due to the fact that the information to be extracted is presented in various formats for human view, mostly in the form of HTML documents (Fiumara 2007; Yang & Zhang 2001),

in contrast to a language that can be easily understandable by a computer (Lam et al. 2008). Also the web pages are dynamically structured and, more importantly, because Web technology is evolving using advanced technologies such as AJAX, JavaScript embedded files, mobile applications and other similar features (Flejter 2011).

Consider, for example, a user who wants to search for information on a 'JavaScript Training' course such as the date, location and the price of the course before he/she can enrol on the Web. The user has to go to the website of each course provider, post some queries, extract the relevant information from the resultant web pages and compare the results manually.

The above is a typical example of how an information extraction is done manually, which requires a substantial amount of effort and time (Lang et al. 2012). Using automation, these steps are no longer required. However, this research is motivated by the fact that the automatic extractor has its own limitations and is usually far less reliable for accurate tuple extraction (Ji 2010, 2006; Irmak & Suel 2005). Furthermore, there will always be errors and gaps in the automatically extracted data that only a human can rectify. This is especially true when it comes to lack of structural information (Cooley et al. 1997), scope and domain change (Dontcheva et al. 2007) and text parsing from natural language (Martin & Sharef 2011). In order to improve the quality and the adaptability of the automatic web information extraction towards managing any changes of HTML webpage structure, it becomes necessary to investigate the issues and limitations of the existing methods. Factors such as level of logical structure, nature of the source, domain and language of the input data determines the quality of the extracted tasks (Pikorski & Yangerber 2013).

The outcome of this analysis of the current state of WIE has led to this research to investigate WIE in relation to the semi-automatic extraction. Semi-

automatic WIE is a process of extracting information with human involvement to identify the relevant information (Laender et al. 2002, Adelberg 1998). Well-presented information is trivial for a human to understand and yet is difficult, if not impossible for a machine, because a machine depends on pre-encoded instructions or previously processed instructions and any ambiguity of text or phrase would be a further difficulty (Lerman et al. 2003). Specifically, the aim of this research has been to develop and evaluate a Teachable Semi-automatic WIE (TS-WIE) system (see Chapter 5) for an industrial application.

The main focus is the ability to teach the system how to extract unknown data patterns that have just been discovered by extending its pattern knowledge base. To have a huge impact on the performance of this system, the trainer (user) must have the domain knowledge and page layout understanding for finding and selecting the relevant data. Trainable WIE has been shown to be effective to support information extraction aiming to maximise reusability and minimise maintenance cost (Chang et al. 2006). Trainable systems can be extended more easily, which requires less domain knowledge. However, the precision and recall performance normally suffers compared to a handcrafted system (Feldman et. al. 2002).

1.3 Research Scope

It is important to note that this research concerns the extraction of information from semi-structured Web sources (HTML documents) from the public domain. The scope of the research is the extraction of information from websites of training courses. In the context of this thesis, ‘websites of training courses’ refers to websites that advertise training courses of some kind within the UK and ‘web pages of training courses’ refers to the web pages containing a course(s) offered by websites of training courses. The course information is chosen here as it is presented in different layouts and various levels of complexity. This is further discussed in Section 5.2.2.

The research problem studied in this thesis is based on the daily operation of a specific organisation which provides complete training solutions to its clients. This establishes a good basis for the study and analysis of requirements and specifications within this training courses domain, in the real-world scenario. The information of interest from this domain is title, date, location, and cost of the course.

Despite the above constraints, it is the aim of this research to discover the diversity of the structure and textual features of web pages describing training courses. The findings from this study contribute to the development of a solution for an effective WIE. Although it is mainly focusing on the course training domain, it is also aimed at producing a generic information extraction solution that is reusable in other domains of similar context or with a different kind of information of interest.

1.4 Subject Domain – Case Study

This research originally studies and proposes a prototype for an industrial application, which requires data extraction from multiple public websites. In this research, the issues and problems are based on the business experience of Apricot Training Management (ATM). ATM is a non-profit organisation located in Loughborough, United Kingdom (ATM 2010). ATM's business is to provide a complete training solution, which is tailored to the specific needs of an organisation or an individual. ATM was chosen as the case study for four reasons:

i. The nature of ATM's business process falls in the WIE field, which is gathering information from the Web, without any prior knowledge of the website's structure.
ii. A prototype WIR/WIE was proposed and developed by Xhemali (2010a) for ATM. The system is an automatic system designed to automatically extract and update the list of courses offered by different training providers into a database. This is a good base for this research to introduce semi automation. The WIR/WIE system is briefly discussed in

Chapter 3. For a detailed discussion of this system, see Xhemali (2010a).

iii. The extraction rules in the collection that are used to assist the extraction process are fixed and manually built. This requires manual and expert updates if new requirements or 'never been seen' structure is involved. Thus this provides an opportunity to extend the extraction to a much wider perspective.

iv. There is an explosive increase of training courses web pages with different levels of presentation complexity, which are likely to be frequently changed or updated.

Despite the promises that the automatic WIR/WIE system can bring to ATM, maintaining the course information collection effectively requires human effort because of the limitations posed by this system (problems of the existing automatic system are detailed in chapter 3). Moreover, with limited lifespan of some courses, the organisation is now facing the problem of keeping the course information up-to-date. However, this thesis proposes an extensible model to enhance the automatic extractor, specifically the quality of extracted information and wider coverage of information extraction.

1.5 Research Aim, Research Questions and Hypothesis

The research aims to provide an efficient mechanism that learns data patterns to extract the required information, which is dynamic, from the Web sources at an acceptable time and human effort by:

1. Developing a solution for WIE of training course data that builds on and improves solutions derived in earlier research, using Apricot Training Management (ATM) as the case study.
2. Showing how GP methods can contribute towards enhanced methods of WIE.
3. Identifying generic benefits from the methods developed that could be used by a range of other organisations.

The above aims are achieved with the following objectives:

1. Review the problem requirements of ATM's training courses for the purpose of understanding the characteristics of the problem so that solutions to the problem can be proposed and developed.
2. Carry out a literature review to show that:
    a. GP is an appropriate method to develop the required WIE solution
    b. a semi-automatic method requiring expert human input is an appropriate approach to develop a better WIE solution.
3. To justify the research methodology, review the earlier research by:
    a. Mark Withall (2003), to evaluate how his work could be used as a foundation for developing a suitable WIE software tool.
    b. Daniela Xhemali (2010a), to identify areas where her work could be improved, but that her work would make a good foundation for an improved WIE solution.
4. Develop and test the software required using a combination of the GP method suggested by Withall and XML-based grammar suggested by Xhemali. This is presented in Chapter 4 and Chapter 5.
5. Apply, develop and test a TS-WIE system using the method used in (4). This also includes regular expression learning given an initial regular expression and labelled example. The method is presented in Chapter 6.
6. Evaluate the solution with experiments on web sources from the chosen domain (based on ATM case study).
7. Review the solution and methods used in ATM case study to identify generic benefits from the methods used that could potentially be used by other organisations.

The research intends to answer the following research questions:

RQ1 - What are the common and distinctive design characteristics of the training course web pages?

RQ2 - What area of the automatic extractor can be enhanced that would improve the quality of the extracted information from

"never seen before" web pages?

RQ3 - How does semi automation and improved GP method support the automatic extractor system?

RQ4 - How much change needs to be made to the extractor to make it adaptable to other domains with similar or different extraction attributes?

The research hypothesis to be tested in this thesis is shown below:

"A Teachable Semi-automatic Web Information Extraction System (TS-WIE) with human supervision helps to achieve high quality extraction and may increase adaptability to a wider scope of domains compared to an automatic Web Information Extraction System alone".

Table 1 below shows the seven objectives that have been developed from the research questions, together with the method by which each objective will be attained and the research question that it will answer.

Table 1. The objectives in relation to the methods and the research questions.

OBJECTIVES	METHOD	RESEARCH QUESTION
#1 – understand the characteristics of the training course problem	Requirement Analysis to identify current issues and problems with extraction of information from web documents (using ATM as a case study for the solution)	What are the common and distinctive design characteristics of the training course web pages?
#2 – review related work and potential methods for the solution	Literature review	What area of the automatic extractor can be enhanced that would improve the quality of the extracted information from "never seen before" web pages?
#3 – justify the research methodology	Literature Review of earlier research	
#4 – Develop and test GP software	Computer Program Evolution (CoPE) and Regular Expression Evolution research (REGEXEV)	How does semi automation and improved GP method support the automatic extractor system??
#5 – Develop and test proposed WIE system	Conduct semi-automatic WIE (TS-WIE) research	
#6 – Evaluate solution	Experimentation using ATM data	
#7 – Identify potential use of solution/methods to other domain/organisations.	Review solution to identify generic benefits	How much change needs to be made to the extractor to make it adaptable to other domains with similar or different extraction attributes?

1.6 Significance of the study

The first Message Understanding Conference (MUC) was initiated in 1987. The establishment of MUC (MUC1 – MUC7) to stimulate research in this area shows evidence that IE is an important technology with a promise of improved quality of extracted information and adaptation to new environment.

An ideal situation for an effective WIE is when all the structures of the HTML web pages comply with the W3C web page design standard but this is impossible to realise (Crescenzi & Mecca 1998), especially when it involves handcrafted pages or modification made by humans with minimal technical training, compared to those created using scripts (Cohen & McCallum 2003a). Moreover, (Gibson et al. 2005) estimate that 40%-50% of the web pages contain irrelevant data and predict that this will increase.

Despite the numerous innovations in WIE, achieving sufficient quality or accuracy and adaptability remains unfulfilled (McCallum & Jensen 2003). Authors, such as, Sun et al. (2011), Eikvil (1999) and Chidlovskii et al. (1997) highlight that some of the key difficulties in data extraction concern the diversity of the content, sparsely related data and page layout. A recent study by Flejter (2011) shows that although modern WIE systems in general are capable of handling a diverse complexity of web document, new and rapid development in Web technologies such as AJAX, CSS and JavaScript, continues to bring new challenges that need to be handled. Flejter makes the point that out of 336 challenges, only 195 have been addressed by over 40 information extraction systems. This shows that the area of information extraction is still evolving.

This study is significant in two ways; academically and practically. From the academic perspective, this study underlines the unique characteristics of semi-structured web pages for the benefit of modelling the web information extraction algorithm, the application of simple extraction techniques; combining textual and structural learning method, and uncovering the benefit of evolved extraction patterns in this area.

From the practical perspective, this study provides a motivation for system implementers to consider semi-automation as humans still outperform machines in picking out relevant information from a mass of data. Moreover, to sustain the quality of the extracted information and the diversity of the domain, the information extraction system needs to be taught about new discoveries or new knowledge.

Evidence from this study shows that not only does it provide a solution for the organisations which provide customised complete training solutions, but it is also applicable to other business nature, for example product listing and books retail, with provision that information presentation, unlike newswire or blogs, has some kind of structure. Furthermore, only minimal adjustment is necessary to suit the requirements. Finally, the study aims to make recommendations based on the experimental evidence for further research.

1.7 Thesis Layout

The thesis consists of 6 chapters including this introductory chapter. An overview of the remaining five chapters is as follows:

Chapter 2 presents the literature review on the current development issues and breakthroughs in the discipline of Web Information Extraction (WIE), including artificial intelligence techniques to solve optimisation problems. The domain of optimisation problems is discussed from a general perspective before focusing on the design domain, particularly on the evolution of regular expressions to capture the required information. Finally, the research gap is presented leading to the formation of the objectives, research questions and hypothesis.

Chapter 3 describes the earlier work relevant to the development of the TS-WIE system. Before developing the TS-WIE system, it is essential to study ATM's automatic WIE system to identify its purpose, requirements and

features. This study aims to justify this research by highlighting issues and problems of this automatic system. Prior work of the evolution of complete software systems, which inspired this research, is also presented.

Chapter 4 (Phase 1) discusses the work on the evolution of a complete computer program with the majority of the works involving experiments to find a suitable solution to the problems and issues discussed in chapter 2 and 3. Based on this work, two papers have been published in *Proceedings of the 4th International Conference on Evolutionary Computation Theory and Applications*, Spain, October 2012 and in *Proceedings of the 7th WSEAS International Conference on Computer Engineering and Applications,* Italy, January 2013. This work is then followed by the evolution of regular expressions.

Chapter 5 presents the application of similar method to Regular Expression domain. The works in both Chapter 4 and this chapter are the initial works which are necessary to assist the earlier development of the TS-WIE system, specifically optimising the evolution of regular expression patterns used for the extraction.

Chapter 6 (Phase 2) introduces the TS-WIE system's framework and the development. This system requires involvement of a human user to provide assistance in identifying the relevant information presented on a web page. The interaction provides the opportunity for the system to learn and acquire new knowledge for better Information Extraction coverage. The design of the user interface is also included. A method to evaluate the TS-WIE system's effectiveness (in terms of its recall, precision and F-measure rates) is also outlined. Finally, the experiment on the regular expression evolution technique is repeated and the results of before and after grammar changes are compared to find out the impact of the incremental grammar by the TS-WIE system to the performance.

The final chapter, Chapter 7, draws the conclusion with the summary of the research findings and highlights the significant contribution of the research to the WIE field, the limitations of the research and the proposal of some areas for further study.

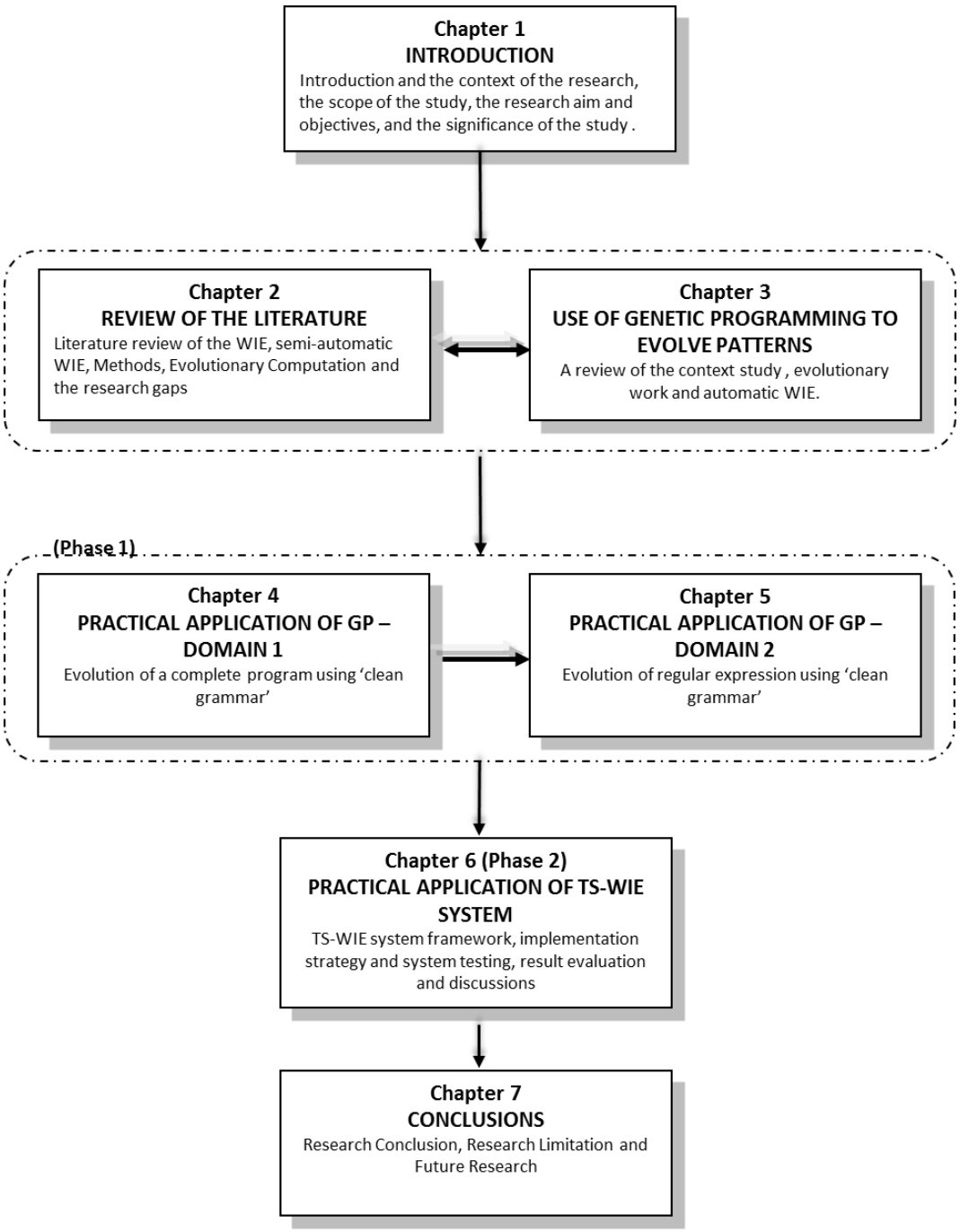


Figure 1. Thesis Structure

1.8 Chapter Summary

The growing complexity of the Web demands Information Extraction tools which are able to handle the so-called information overload. The Web Information Extraction (WIE) evolved a regular expression to automate the extraction tasks and offers the potential of a significant improvement in the quality of extraction. However, for this WIE to be successful on new discovery of unseen data structure, its knowledge base containing extraction rules needs to be continuously updated.

This chapter provides a brief introduction to the research including the research background, research aims and questions, the research contributions and the case study. The research provides a system to solve the issues of WIE faced by the course training domain in general and specifically by Apricot Training Management. In order to understand how improvement can be made to the quality of information extracted, it is important to analyse the design of the existing automatic WIE system including the design of its database. The result of this research is a generic, pluggable system called the TS-WIE system, where human intervention plays an important role in providing the set of training examples. The next chapter reviews the background literature of WIE and Genetic Programming in more detail.

Chapter 2

Review of the Literature

2.1 Chapter Overview

This chapter provides an overview of the related literature for the scope of work studied in this thesis. The objective of this chapter is to investigate the theoretical potential of GP towards enhanced methods of WIE solution and the usefulness of semi-automation to enhance the automatic approach. There are two areas in focus; Web Information Extraction and the use of Evolutionary Computation (EC) to evolve patterns for the extraction process. The first part of this chapter generally discusses the origins of IE and how it has evolved over the years and in particular the important area of Web Information Extraction. Various extraction tools used for extracting information from the Web were investigated including wrappers and data extraction rules.

The second part of this chapter discusses the relevant, current practices in EC, specifically Genetic Programming (GP) with particular attention to some new methods that extend the earlier standard GP. GP often provides a combination of features and rules, which require a knowledgeable programmer, and are sometimes even difficult for the programmer to think of and write. Moreover, this combination appears to effectively produce better results (Gordon et al. 2006). However, if this method is not handled carefully with proper control of features and parameters, especially if the search space is huge, it could become destructive and cause the algorithm to have an exponential time complexity (Poli & Langdon 2007). This section emphasises how GP can be used to evolve computer programs to solve problems and to generate regular expressions for data extraction, justifying the second thesis objective, i.e., why this research considers GP an appropriate method to develop the required WIE solution.

This chapter is closed by the concluding remarks including the summary and conclusion.

2.2 Web Information Extraction (WIE) and its building blocks

There is a great deal of discussion in the literature to suggest that individuals and organisations are increasingly relying on WIE to support the decision-making activities and meet the organisation's objectives respectively (e.g. Ferrara et al. 2013; Sunny & Sundar 2013; GAC 2012; Kenjibriel & Akbar 2012; Krishnamurthy et al. 2008; Sarawagi 2008; Baumgartner et al. 2005; Kuhlins & Tredwel 2002; Ciravegna, 2001; Andersen et al. 1992 and Porter & Millar 1985). A WIE system is typically implemented to provide key information to support the organisation's processes and reduce the employees' time as a result of inefficient searches (Gao et al. 2013; Sarawagi 2008; Bhide et al. 2007; Zhu et al. 2007; Popowich 2005). Moreover, WIE (HTML title extraction) also provides results that are useful to enhance Information Retrieval (Xue et al. 2007; Zhang et al. 2002; Cutler et al. 1997).

The main task for WIE is always the question of how to identify and gather information from a semi-structured or unstructured collection containing potentially relevant information and transform it into a suitable form that can be automatically queried by other applications in future. The four phases of WIE described below are the consolidation of commonly established processes described in the literature :

1. Collect web data including web content, page structure and application data. In this thesis, as in several other works in the field of IE, crawlers are used to retrieve the relevant documents from the intended domain. More specifically, a customised crawler is used rather than Google search engine and this will be presented in Chapter 3.

2. Pre-process web data to transform it to a format compatible with the analysis technique, such as cleaning data abnormalities, filtering 'noise' and correcting missing links.

3. Analyse web data to define patterns and statistical correlations between web pages and user groups using techniques such as data mining and machine learning.

4. Recommendations presented to the user based on the results of the previous analysis.

Looking at the pioneering works of IE - the predecessor of WIE, IE was first initiated by Schank in 1975 for Natural Language Processing (Schank 1975). In the mid-1980s, the first commercialised IE system called JASPER was produced (Andersen et al. 1992). JASPER was developed by the Carnegie Group for Reuters Ltd. providing the earnings and dividends information extracted from company press releases published by PR Newswire, in a form of Reuter's news story for use by financial traders.

To promote research interests and evaluate the state-of-the-art in IE, the first Message Understanding Conference (MUC-1) was initiated in 1987 with support from DARPA. MUC has become the major reference source in the IE field (Appelt 1999) and up-to-this-date, the conference has produced proceedings, using training in the domain of airline crashes and events launches. Early extraction tasks were focused around named entities identification from natural language text such as people and company names and their relationship (Sarawagi 2008). Communities' requirements are manifold such as shopping comparison and financial applications, and now IE techniques have evolved considerably to address these requirements and adapt to different topic domains. The massive growth of these information/documents on the Web and its impact on the search time and effort, in the late 1990s, work on Web Information Extraction (WIE) has attracted some interests from researchers.

This thesis is concerned with the task of WIE from the web pages that are rendered for human view. Most of the web pages today are developed as Hyper-Text Markup Language (HTML) documents. The categories of HTML pages are discussed below.

Researchers have categorised the contents of these pages into three main types, which are structured data, semi-structured data and unstructured data. However, the definitions of this terminology are slightly inconsistent (Chang et

al. 2006), therefore, in the context of this research, the following are applicable:

- **Structured data.** Some researchers define structured data as data which has some kind of structure/format, or schema, normally from structured data sources e.g. databases (e.g. Lam et al. 2008; Fiumara 2007; Arasu & Garcia-Molina 2003). Examples of these data are on-line stock quotes and weather reports.

- **Unstructured data** is defined as free texts, which have no data model (Chang et al. 2006; Fiumara 2007). Sources include blogs, news articles and memos.

- **Semi-structured data**. Researchers like Lam et al. (2008), Fiumara (2007) and Chang et al. (2006) define semi-structured data as anything in between structured and unstructured data where data are usually expressed in tables, itemised or enumerated lists. A large number of HTML-based web pages are categorised as semi-structured. This is because the data expressed in those pages do not conform to any formal structure. Data are rendered using implicit underlying HTML tags because data are usually mixed with tags and layout formatting. This thesis explores the semi-structured data presentation, which contributes to the choice of methods for the WIE solution.

One of the most challenging and interesting tasks in WIE is identifying all the important information out of all the irrelevant ones. Researchers like Abolhassani et al. (2003) and Kaiser & Miksch (2005) made a point that the design of WIE system will affect the quality of the extraction and the information identification algorithm must be: (1) accurate – it should be able to identify the same information accurately even though it is presented in different forms; (2) reliable – it should produce the same result without being highly dependent on specific hardware and software (e.g. browser or platform); (3) adaptable – it should be able to tolerate changes in different environments (e.g. changes of web page structure, changes of information position).

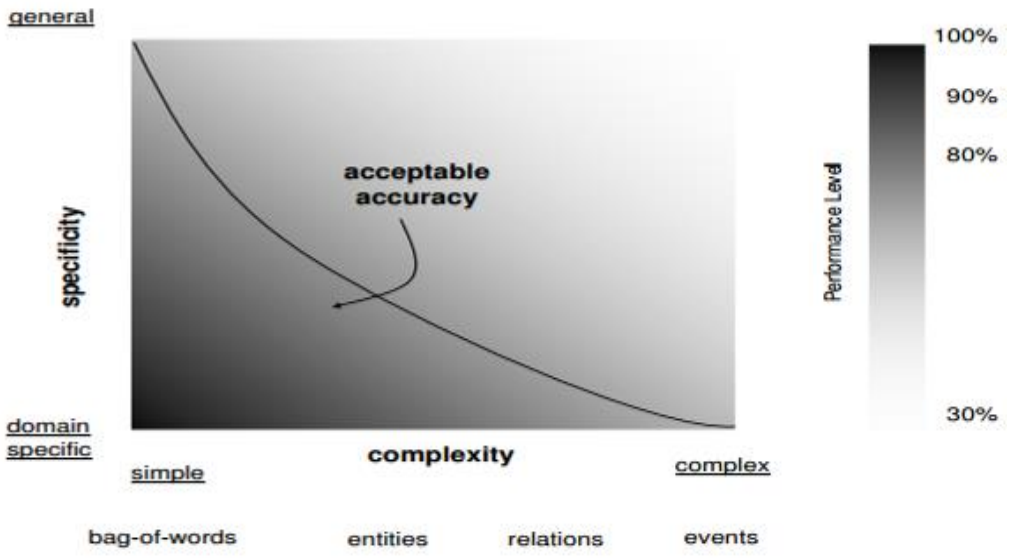


Figure 2.1. Performance trade-off relative to specificity and complexity (source Cunningham, 2005).

Figure 2.1 shows the relation between specificity and complexity of an IE in terms of achieving quality information. High accuracy can be achieved if the data to be extracted are complex but the domain is specific. Or, if it involves extraction from open domain and the data to be extracted are simple, then a more general algorithm is needed. The simplest data to extract is the textual strings from natural language in a domain specific environment, such as names of persons or organisations (Normand et al. 2009; Alfonseca et al. 2006). More complex scenarios involve entities, entity relations and events extraction with ambiguous records in free text (Pikorski & Yangarber 2013; Hobbs & Riloff 2010) such as capturing who, what, where, when and why from the terrorist acts article in the social websites.

It would be much easier to design an IE system with prior knowledge of the web pages structures or have a special pre-arrangement of the kind of information that will be provided, which could be accessed through protected links such as API or web services, as in the case of GoCompare.com[1] and some other similar comparison websites (ShopBots). If the company that they are dealing with decided to make some changes to the document layout or content it would have less effect on these comparison websites than it would

[1] A financial services comparison website that quotes insurance features and prices from its registered suppliers.

on individuals.

In an ideal situation, an automatic extractor could solve this problem and work effectively if the data are annotated with adequate labels. However, due to the absence of standards in the structure of the web pages and limited appropriate resources (tools and experts) that can be used in labelling training data, it is unlikely that an automatic extractor could function properly every time. Therefore, the focus of this thesis lies in the extraction from the web, investigating how an automatic extractor can gain advantage from human supervision while keeping this supervision to a minimum.

The scope of this thesis concerns a case study of WIE for a service-providing (course training solution) organisation, where the extracted information is essential to support the business events.

2.3 WIE to support organisations' events

Growing competition forces organisations, especially those dealing with customers such as financial institutions, insurance companies, and ATM in particular, to acquire valuable information in a timely manner. This means delivering on-target solutions that achieve satisfaction to the customers' expectations. The value of information varies from one organisation to another, ranging from information related to customer profiles to competitors' activities. WIE is not only a necessary part of supporting business but is also necessary to survive in the global industry; for example the ability of a company selling products to acquire and monitor the products pricing of a rival company will provide an opportunity to offer a competitive price or price comparison for marketing purposes.

Knowledge workers are estimated to spend around 15–35% of time searching for information, but more than 50% of the time they are unsuccessful (Feldman 2004). Search engines like Yahoo!, Google and Bing are normally used to facilitate information searches on the Internet. Based on data

released by comScore, over 66% of searches conducted globally during December 2009 were on Google, and in September 2010 in the UK alone, 43.1 million people conducted at least one search per day. According to the 2013 survey by the Office for National Statistics (2013) about 31% of adults (aged 16+) in the UK searched the internet for information on education and course training, which sees an increase of 6% from 2005. This shows that this domain is still in demand. Based on this statistic, it is also noted that a higher level use of the internet is finding information about goods and services after the activity of sending or reading email. Faced with the problem of information overload, using these search engines to find and access the desired training courses is too time consuming. For example, searching 'course training UK' in Google would yield about 551 million links, Bing returns more than 95 million results at time of writing. Going through each web page, one after another and extracting the relevant course information would put a heavy burden on the user.

WIE is essential for organisations due to the following factors, which demand the efficiency and effectiveness of the WIE systems:

1. **Information value.**

   Data obesity or data explosion has appeared to be one of the problems faced by individuals and organisations (Martin 2005). However, WIE provides a strategy to shed the undesirable from the desired data. The right information is useful (i) to support decision-making activities. This may include product comparison/information/review and customer reviews which can be used for comparison shopping (Etzioni et al. 1997) or overview of trends; (ii) as a source of competitive advantage. The Web has provided an opportunity for individuals or organisations to accumulate data from many different sources and use it effectively to gain competitive advantage such as customer reviews for market forecast.

2. **Speedy access of information**.

   WIE provides communities/users with direct access to the required

information in a structured manner without the need to scan/read through/analyse the webpages to find the information, thus saving a lot of time and effort. It is seriously important that data, especially from the web pages that are frequently updated, for example stock activity and news, are presented to the user in a timely manner.

3. **Populate database**.

WIE makes it possible to capture information from various web sources, transform them into the desired format and integrate them in a single database or XML etc., which can be queried or used for analytical or statistical purposes in the future.

2.4 Semi-automatic WIE

2.4.1 Introduction

There are several excellent survey articles on the many approaches to WIE systems like Ferrara (2013), Sarawagi (2008), Chang et al. (2006) and Laender et al. (2002), which include the wrappers generation classification, the different categories of extraction tasks, the degree of automation and the types of extraction tools. Much of the recent activities on WIE have been stimulated by web page segmentation, which separates boilerplate (advertisements and navigations) to concentrate the extraction from the main content called content extraction (e.g. Lang et al. 2012; Kohlschuetter et al. 2010), thus increasing computational efficiency, open-domain (e.g. Cimiano & Volkar 2005) and web services (e.g. Seidler & Schil 2011; Metke-Jimenez et al. 2011).

WIE systems generally use extraction rules or patterns (Stevenson & Greenwood 2006), which can be hand crafted or automatically learned from training examples annotated by a human expert. This section reviews the semi-automation of WIE methods. The semi-automatic WIE explores the structure, keywords or layout of the parsed HTML web pages, with provision

of training data set. This type of WIE produces a set of extraction patterns or rules that determine what information to extract and how to extract them from the web page based on input from a user collected from a GUI. In this thesis, the patterns are generated based on the grammar definition following the syntax of a specific language (regular expression). The regular expressions are widely used in many programming languages and applications. Explicit research works on solving regular expressions matching problems include Brazma and Cerans (1993) who have considered the efficient identification of regular expression from good examples, and Belazzougui and Raffinot (2013) who have studied the approximate regular expression matching with multi-strings. Another successful application is by Svingen (1998) using GP to evolve regular expressions to recognize several Tomita (1982) regular languages.

Several methodological innovations have helped make semi-automatic WIE possible and practical. The system built by Ashish and Knoblock (1997) uses lexical information (font size), HTML tags, and indentation to guess the structure of a web page. It allows user interactivity to identify for the correct keywords. In 1998, a system called NoDoSE (Adelberg 1998) was developed which attempts to infer the format of the user input for various attributes, relying on the consistent ordering of these attributes in a record. The user is required to decompose the files into a hierarchy of records or lists, identify the regions of interest and specify their semantics.

SoftMealy (Hsu & Dung 1998) assumes that all training examples are available. The wrappers are represented as finite-state transducers. Liu et al. (2000) proposes an extraction system called XWrap. When the user is presented with the system's predictions of the correct data, he/she may interactively teach the system by highlighting the missed tokens or delete the incorrectly extracted tokens. One distinct feature of the user involvement from the previous system is that the user can correct the errors in the system-generated XML-template that describes the structure of the page.

A system which aims to extract data from nested tables requiring the user

involvement to specify a small set of examples is called DEByE (Laender et al. 2002). The key novelty of this approach is that it provides flexibility for the user to specify the examples according to his/her interest. The extraction algorithm of DEByE relies on heuristics based on the position of attributes in HTML tags. Estievenart et al. (2006) introduces Retrozilla, a tool to extract information from a popular on-line movie website (imdb.com). The user of this system is required to select the intended value and provides its label. If unwanted data are also selected by the system, the rule refinement can be done repetitively with the user identifying these unwanted data. A new attempt to automate wrapper generation in a dynamic way is by Jundt and Keulen (2013). They use XPath ranking for finding the attributes of interest based on a small set of examples provided by the user from a number of detailed web pages in bookstores domain.

According to Crespo et al. (1994), a semi-automation approach typically requires some sort of learning mechanism capable of handling both document structural evolution and varying sets of documents. They further define the three phases involved in this learning based approach; *training, processing and feedback*.

Training Phase: In the training phase, the user provides some annotated training examples (positive and/or negative examples). A positive example is a piece of text known to be correct, and a negative example is the opposite. Training examples determine the characteristics of the attributes to be extracted and the system normally learns from these positive examples to produce the extraction rules.

Processing Phase: the algorithm takes the examples and induces the extraction rules (or patterns). These rules are then applied to new inputs. The data that matches with the rules are presented to the user for necessary filtration.

Feedback Phase: the user then trains the system by identifying any incorrect results presented to him/her (called negative examples). This process guides the system to learn the changes and performs a stepwise refinement of the generated rules to improve the quality of the patterns it generated. The

processing and feedback phases are repeated until a satisfactory performance is achieved.

2.4.2 Wrapper

In a traditional approach, a human expert is needed to handcraft a wrapper using specialised programming languages for each website to recognise the information of interest among other uninteresting information, such as markup tags and transfer it to some format such as database, spreadsheet, XML. A wrapper, in its simplest definition, is a generated or coded program used for extracting important information from a particular source.

In the context of WIE, a Web wrapper is defined by Ferrara et al. (2013) as a procedure (one or different classes of algorithms) used to find data from the Web as required by a human user, extract them and transform them into a relational form for further processing. Because the ‘wrapper’ used in the extraction task in this research is specifically represented in a form of regular expression pattern, this ‘wrapper’ is referred to as ‘regular expression pattern’.

A wrapper normally relies on a set of extraction rules to perform a data pattern matching and is normally created for each information resource. Wrapper generation normally involves four processes; retrieving training pages, generalising of the extracting rules, extracting data and transforming output into structured data. Generalising rules is done by either replacing tokens with general token feature (e.g. wildcard) or dropping the redundant tokens (Sarawagi 2008). One of the drawbacks of using a wrapper approach is when the layout of the web page changes, which it is not programmed to handle, and inaccurate results may be produced.

In an attempt to overcome the shortfalls of manual crafting of a wrapper, inducing a wrapper has been extensively studied and been put forward in the literature e.g. prefix-suffix pair (Kushmerick et al. 1997), finite-state automaton (Muslea et al. 1998), XPath (Anton 2005; Myllymaki & Jackson 2002), Elog rules (Baumgartner et al. 2001) and XML (Liu et al. 2000). Chang et al. (2006)

surveyed these works and categorised them into four classes according to the degree of automation; manual, unsupervised, supervised and semi-supervised as shown in Figure 2.2. Eikvil (1999) provided extensive information on wrappers and wrapper generation.

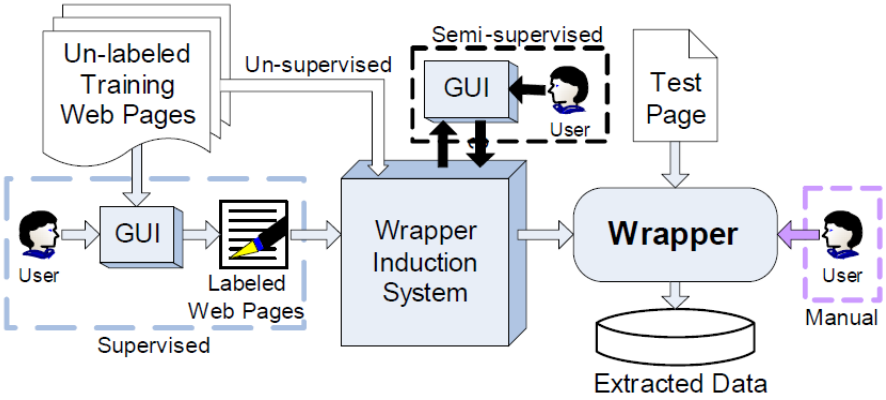


Figure 2.2 General view of Wrapper Induction System illustrating the four categories of system's degree of automation; manual, unsupervised, supervised and semi-supervised (source Chang et al. 2006).

Manual Wrapper – This type of wrapper generation merely supports the user to handcraft the specific wrapper. TSIMMIS by Hammer et al (1997) is one of the first systems to implement this method. This system takes input from a programmer using a sequence of commands consisting of the data location and how the data are to be put into objects and in return, the system outputs the desired information. Unlike TSIMMIS, WEB-OQL by Arocena & Mendelzon (1998) is used as a query language and it provides hypertree (labelled ordered trees with three attributes; Tag, Source and Text) for semi-structured data such as a relational table and a directory hierarchy. This system requires a 'select-from-where' query format from the user (a programmer) to extract the required information. In the same year Minerva was proposed by Crescenzi & Mecca (1998). This system uses a declarative grammar-based approach incorporating procedural programming features for generating wrappers. The grammar is defined in EBNF containing a set of productions to define the structure for each source document. A different approach was introduced through W4F by Saiiuguet & Azavant (2001). W4F is developed as a Java toolkit to build a wrapper and it consists of three

layers; retrieval, extraction and mapping. Extraction rules are expressed manually using a HTML parse tree path to locate a specific data.

Although these systems can achieve accurate extraction, writing the wrapper requires the skill of the knowledge engineer, i.e., a person, with a substantial programming background and a good understanding of extraction rules or is linguistically competent to develop robust extraction rules by hand (Wong 2012). However, creating rules by hand is difficult and time-consuming (Eikvil 1999; Riloff 1996) and sometimes incomplete, inconsistent, or even partly erroneous (Suwa et al. 1982). Riloff estimated that approximately 1500 person-hours of effort are required to write the dictionary used for the data extraction (Riloff 1993). Crescenzi et al. (2001) added that this difficulty not only concerns the wrappers generation, but also their maintenance. Performance of this manual system depends highly on the competence of the experts.

Unsupervised Wrapper - To reduce the burden of writing the wrappers an unsupervised extractor is introduced. This is an automatic extractor to discover the data of interest and similar data items in the same page or multiple pages of a single website or multiple websites. An example of a system implementing this approach is RoadRunner (Crescenzi et al. 2001). RoadRunner uses the ACME matching technique to compare HTML pages and, based on their similarities and differences, a wrapper is generated. EXALG (Arasu & Garcia-Molina 2003), like RoadRunner, is a page-level extractor based on template and schema deduction using multiple web pages from the same website.

The unsupervised approach above uses a number of general assumptions about the data of interest to increase the extraction rate. Although this type of extractor does not require human involvement, it does not always extract accurate information. This is because they are highly dependent on well-formed documents and because it is lacking in precision, the extracted data might need other applications such as data cleansing and data integration, before it is usable by the intended application. Liu et al. (2003) demonstrated

that these methods produce poor results in relation to this perspective.

Supervised Wrapper – This wrapper generator is also known as **Wrapper Induction (WI).** Many WI approaches use Machine Learning to learn a generalised pattern to build extraction rules. Systems based on wrapper induction allow the provision of positive and/or negative examples through the system's GUI. This kind of supervised system is largely dependent on the user, who should be the domain expert to identify and label examples that will be representative of the actual setting. Chang et al. (2006) made a point that supervised approaches extend well to non-template pages, provided that users choose proper features for the extraction rules.

A system called WHISK was introduced by Soderland (1999). This system can handle extractions from a wide variety of documents ranging from structured to unstructured. It uses a syntactic analyser and a semantic tagger to learn text extraction rules automatically. However, a user is required to provide positive training instances to guide the creation of extraction rules and test the performance of the proposed rules. Another system called STALKER was developed by Muslea et al. (1999). This system performs extraction from a wide range of semi-structured documents by describing the structure of the page in a tree-like structure called embedded catalog tree (EC tree), consisting of leaves (attributes) and nodes (tuples). The user is required to provide a set of training examples, each containing a sequence of a token and an index indicating the start or end of this token. An interesting approach to automate the labelling process, thus reduced human involvement was introduced by Kushmerick (2000). He developed a system called WIEN which is successful on ordered attributes in a data record, especially nested data. However, this automation restricts the capability of the system to handle missing attributes, nested structures and variation of attributes, thus adaptability to the real web environment is difficult to achieve.

Based on the literature, inductive learning poses several problems. The training set may not fully represent the template of all pages, thus poor performance can be seen on different template pages (Crescenzi et al. 2001).

An attempt to solve this problem is by labelling more pages. However, manual labelling of data are labour intensive and time consuming (Zhai & Liu 2005). Zhai & Liu also pointed that another problem is that wrapper is data source specific. This means any changes to the source may cause the wrapper to be unusable. In this case, the same labelling process needs to be repeated for data, which has a different pattern, thus maintenance is difficult.

Recent systems (hybrid systems) introduce learning-based wrapper generation, such as IEPAD and OLERA, to try to reduce this extensive data labelling. Unlike a supervised approach, which insists on exact data, this approach only requires rough examples, and is elaborated next.

Semi-supervised Wrapper - In contrast, a semi-supervised extractor aims for a lighter involvement of the user. It only requires the user to identify the relevant data, which is referred to as the training data, in a record-level extraction task context, either in multi-page or a single page website to help the automatic generation of the extraction rule. These training/labelled examples will be used as seeds. The involvement of the user to help the system in the generation of extraction rules (including correcting) has resulted in a more efficient extraction. However, most systems of this kind lack the ability to generalise rules and to automatically and dynamically extend their rules. Similarly, in this thesis, only minimum human engagement is required during the training provision session and the generation of wrappers relies on machine learning techniques (Chapter 6).

XWrap (Liu et al. 2000) is one of the popular semi-supervised systems. It provides six pre-defined sets of data heuristics to be selected by the user to locate data objects in a specific source. Another approach is the IEPAD system, developed by Chang et al. (2003). To discover extraction rules to extract data from the relevant web page, IEPAD defines and generalises patterns from the HTML tags. It relies on a PAT tree or a suffix tree to find repetitive patterns from the page or from web pages with similar structure. The requirement of repetitive patterns is based on the assumption that on a web page, the same template is often applied to multiple data records or at

least two web pages from a website are similarly structured. This means this technique only supports multiple records extraction. OLERA (Chang & Kuo 2004), which is designed with visualisation support, on the other hand, only requires the user to specify a simple annotation of the block containing a record as an example to produce the extraction rules. The system can discover other similar records automatically and presents all the extracted records for attribute labelling. Unlike IEPAD, OLERA can handle extraction from pages containing single data records. Another system using semi-supervision is Thresher (Hogue & Karger 2005). Thresher, like OLERA, requires the user to specify examples and label them. Thresher uses tree edit distance to create a wrapper.

2.4.3 Extraction Techniques

Although recent years have seen a rapid growth of multimedia content, in most of the typical HTML documents the majority of the important area (the main content) is still covered by text (Kohlschütter & Nedji 2010; Levering & Cutler 2006). Early WIE approaches focused only on document text, while more recent ones exploit the HTML structure and entities relations. According to Baumgartner et al. (2009), there are four main approaches to define wrappers to identify the relevant text content: *Functional approach* (manipulating DOM tree), *Logical approach* (using predicates defined by expressions or programs such as XPath), *Automata Theoretic approach* (manipulating tree automata) and Textual or Lexicon approach (using string pattern matching). Their performances and limitations reported in the literature vary.

Generally, the design of extraction methods aims to respond to what kinds of data can be found in specific sources and where to find them precisely. There is a large body of related work in information extraction manipulating Document Object Model (DOM) tree such as Breuel (2003), Cai et al. (2003), Gupta et al. (2003), Reis et al. (2004), Sun et al. (2011) and Omer et al. (2012), also known as tree-based (Ferrara et al. 2012). To be successful, the

information extraction using this DOM tree approach typically relies on the rich HTML structure.

DOM Tree Structure

DOM is an Application Programming Interface (API), consisting of a standard set of objects, which defines the logical structure of documents (HTML/XML) on the Web and it is usually used as a means to manage the documents (add, edit, delete). DOM is manipulated and assessed through its nodes (an ordered tree containing elements, text, attributes and comments). Only the element nodes and tree nodes matter to the WIE implementation in this thesis.

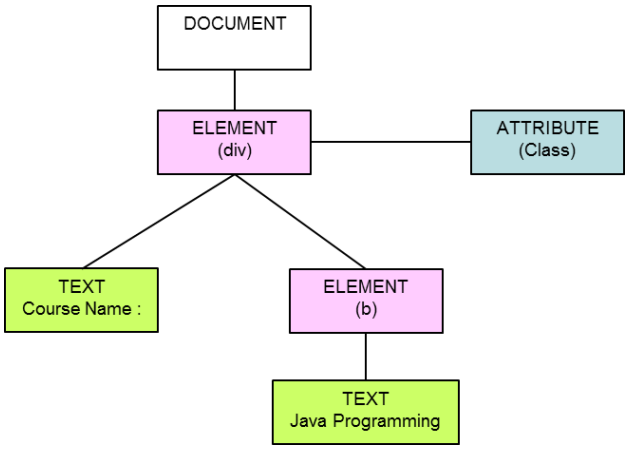
When a web page is rendered, its DOM structure will be automatically produced and represented in a hierarchical manner. The structure is made up of nodes; tag nodes and text nodes. A tag node may be simple such as <table> and it may contains HTML attributes <table id = “courseTable” class = “courseClass” width = “100%”>. Table 2.1 shows the different DOM tree structure to display information in the HTML document.

Table 2.1. Sample HTML structure to display the data.

Item	Example HTML Structure
Single Table	<table><tr><td> 09 November 2011</td>
Nested Tables	<table><tr><td><table ><tr><td>Basic Accounting</td><td>12 Nov 2011</td></table>
division	Accounting
paragraph	Intermediate Accounting
list	Date : 12/12/12

In this study the DOM tree is one of the essential structures to define the physical location of the course attributes within the web page. Figure 2.3 shows the DOM tree presenting the course name in a HTML web page. This

is coded as Course Name : Java Programming . The HTML tag may contain an attribute and is normally followed by the data content e.g. Java Programming.



Course Name : Java Programming

Figure 2.3. DOM representation of HTML tags.

DOM structure is important to discover the location of the information on a web page. Matching this structure poses two challenges; inconsistent pairing of the tags and creating a generic path.

Inconsistent pairing of the tags. In principle, the data or text is presented in between a pair of opening and closing tags such as <table></table>, , and . However, this is not always the case as some of the web pages are created using text editors as opposed to integrated development environment (IDE) or ready-made templates to reduce syntax inconsistencies, thus some of the tags are intentionally not included as they do not have any impact on the live web page. This is because in HTML documents, it is not mandatory to have the closing tag for some tags to be processable by the computer, like <P> and <DIV> and also the tags are not case sensitive.

To ensure the consistencies of structure, messy markups in the HTML document needs to be fixed using HTML cleaner tools such as HTML Tidy (Raggett 2012). This is because properly written HTML will render better and faster than HTML with errors. Moreover, a valid and standard compliant HTML

document generates consistent DOMs that can effectively be manipulated by the scripting software.

Creating a generic path. The content of web pages from various websites when viewed using the browser will look exactly the same although the underlying tags used are different e.g. some websites uses tags to present the data in tabular manner and some use <table> tags to achieve the same presentation. This implies that a pattern which is good for a particular web page will not be useful on another which looks similar to the human eye.

Ideally, to reach an element on the page, an absolute path is required. For example, if the element is in the second column of the second row in the first table, the path represented in JQuery notation is html>body>table:eq(1) >tr:eq(1)>td:eq(1). Absolute path is excellent to reach the data if its location on the web page is known in advance. However, this technique is too rigid and is structure dependent. Moreover, it will fail if this structure changes. Thus a more generic path is more favourable and regular expression could be used. jQuery and Regular expression are introduced in the next section.

Figure 2.4 illustrates node tree and relationship between the nodes. These nodes are useful for the extraction method applied in this thesis; in particular, it is used to capture a single attribute instance as well as the multiple attribute instances (parent-children and siblings relationship), which are similarly structured.

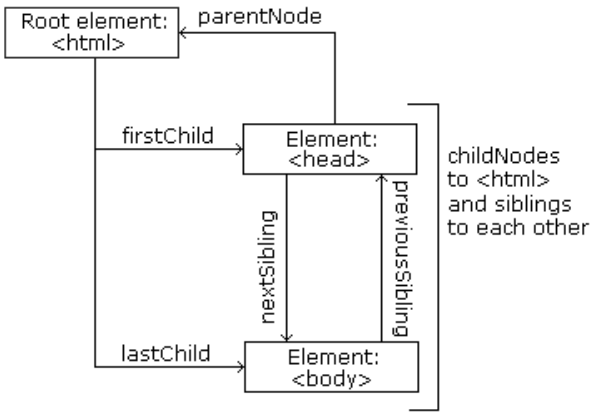


Figure 2.4 HTML DOM node tree and relationship between nodes (source w3schools, 2013).

Introduction to jQuery

jQuery is a light weight variant of the JavaScript library, used to control HTML events, animations and other interactions on a web page. It is a client side scripting, which provides an easy and fast way of HTML DOM traversing and manipulation. For instance, in Figure 2.5, the title can be assessed simply by using jQuery path 'html>body>table>tr:eq(1)>td:eq(1)'.

```
<HTML>
<BODY>
<TABLE>
<TR><TD> </TD>
      <TD>Course Title</TD>
      <TD>Price</TD>
</TR>
<TR><TD>1</TD>
      <TD>Introduction to jQuery</TD>
       <TD>450GBP</TD>
</TR></TABLE>
</BODY>
</HTML>
```

Figure 2.5 Basic HTML structure

The number in the brackets (predicate) is used to find a specific node or a node that contains a specific value in the tree. Note that the index of the html tag starts from zero and it is common that this lowest index is not specified. The *:eq(n)* - where n is the index - after the HTML tag name, specifying the exact location of the selected item. The '>' symbol separating the node name (html, body, div, table, tr and td in the example) indicates *all direct elements* from the parent (or root in this case). Without the '>' symbol, *all elements* that are descendants of 'html' will be selected. Figure 2.6 shows an example of a jQuery path.

jQuery path : html>body>table>tr:eq(1)>td:eq(1)		
Html	➔	The html is the root of the structure
body	➔	The main body of the document where all of the content is placed
table	➔	Refers to the first table in the second div
tr:eq(1)	➔	Refers to the second table row of the first table
td:eq(1)	➔	Refers to the second column of the second row of the first table in the second div. This is the detailed node which holds the text that defines the lexical pattern.

Figure 2.6. Example of jQuery path and their representation.

XPath and jQuery

In WIE, XPath has been a method of accessing the portions of a DOM (DOM nodes), for retrieving the relevant information, and its employment as an extraction technique on the web page has been largely exploited in the literature. It is a powerful query language for many HTML parsers to select a particular element and it is also commonly used in XML documents (Abolhassani et al. 2003).

Xu and Dyreson (2007) proposed an approximation path expression called ApproXPath with an assumption that XPath is ineffective when there are irregularities in data and schema. This approach differs from the exact XPath expression that it can tolerate a web page containing structural errors, and handle a number of user-specified content. Estievenart et al. (2006) prove that their system called Retrozilla, which method depends exclusively on the HTML structure, failed on part of sentence extraction (text node contains more textual information than just the instance of the attribute). The same limitation applies when there is more than one attribute or multiple instances of attributes separated by commas. Another method that can perform a similar task is jQuery path.

Although XPath and jQuery share the same fundamental purpose, jQuery has a number of advantages over XPath, such as simplified code (smaller file and

faster loading), cross-browser compatibility taking full advantage of JavaScript, supports AJAX and it is separate from HTML mark-up, thus it does not meddle with a page's existing HTML and therefore, DOM handling is easier. Because of these reasons, jQuery is chosen for this research.

One of the major drawbacks of using HTML structure is that the algorithm is not flexible. It cannot be used on other dissimilar HTML structures. Furthermore, the majority of wrappers are highly dependent on the tree structure of a given web page, thus when the layout and code of web pages change, they become obsolete (Baumgartner et al. 2009). A robust wrapper should be able to 'auto-heal' to adapt to such changes.

Several researchers like Negm et al. (2013), Ferrara et al. (2012), Muslea et al. (1998) and Kushmerick (1997) claimed that the manipulation of HTML hierarchical structure (DOM) to learn perfect or nearly perfect extractors is able to achieve high level of accuracy in certain domains. However, some recent studies argued that DOM manipulation alone is insufficient to provide definition for the discovery of the important data (Cohen & McCallum 2003a). Therefore, suggestions of more viable approaches were proposed through combining HTML hierarchical structure manipulation together with input features, visual cue and/or visual 2D (e.g. Gatterbauer & Bohunsky 2006; Krupl et al. 2005). Input features may include the string properties (e.g. capitalisation, keywords), formatting (e.g. font size, colour and font style), length of text and data type (e.g. number, string) and visual cues look at the grid of the document.

Because DOM manipulation is one of the approaches implemented here for the WIE system, it is not in the interest of this thesis to remove irrelevant content which could destroy the structure of the web page, rather than focussing strictly on fixing the imperfect structure and standardising the HTML tags used, e.g. <italic> is replaced with <i> and – with a '-'. Due to the different structures of HTML web pages, DOM approach is enhanced with lexicon extraction.

Most lexicon extraction concerns extraction of named entities such as person names and location. According to Xue et al. (2007) relying on the title field to extract the title is risky as 33.5% of the HTML documents that they studied have bogus titles; empty title field, 'untitled' title field and duplicated title field. Titles in the bodies of HTML documents are much more reliable as they are presented to the human. Xue et al. (2007) give more details. Unlike named entities, structured entities such as dates and times can often be identified using simple regular expressions (Abolhassani et al. 2003).

Regular Expressions

The course attributes are made up of text (alphabets and special characters). Text has its own pattern, which can be matched with pattern-matching tools such as regular expression, Parsing Expression Grammar and finite state automata. The goodness of the regular expression always depends on data peculiarity, considering various structural representations and the form of data representations.

Regular expression was first introduced by (Kleene 1956) and is an extremely powerful tool to describe the sequence of text patterns. It is applied in many diverse programming languages such as Java, PHP, C++ and C. Regular expression is widely used, particularly in the Unix community as a searching/replacing tool and has been successful in matching data patterns (Muslea et al. 1999; Soderland 1999; Embley 2004; Li et al. 2008; Wu & Weld 2010; Xhemali 2010a; Liu et al. 2010) for various information extraction methods.

Regular expression can be used to match phone numbers, email addresses, HTML tags and other text strings. For example, *[a-zA-Z]+* will match all whole words such as *"html"* or *"Html"* but not *"<html>"* and *"</?[a-z][a-z0-9]*[^<>]*>"* will match any opening or closing HTML tag such as *"<table>"* or *"</table>"*. Generally, it is a pattern notation with various level of complexity that can match against all kinds of text strings (Sun Microsystems 2010, Friedl 2006).

Regular expression may be built from combinations of basic syntax and special character classes (refer to Appendix 1 for the detail). Table 2.2 shows some of the basic notations of regular expression.

Table 2.2. Basic regular expression notation

Character	Description	Usage Example
QUANTIFIER CHARACTER		
.+	Matches at least one preceding character	.+se will match use, course, the course but not se
.*	Matches zero or more preceding characters	.*se will match se, use, course
.?	Matches one or none character.	Prices? will match price and prices
.*?	Matching the preceding character zero or more times – non greedy.	c.*?e matches ce in the word 'celebrate' (without ? this would match celebrate)
+?	Repeat matching the preceding character one or more times – non greedy.	c.+?e matches cele but not ce
{min} **{min,}** **{min, max}**	Matches exactly the minimum of occurrences Matches at least minimum of occurrences. Matches at least minimum and not more than maximum of the preceding character.	Fe{3} will match feee but not fe, fee fe{2,} will match fee, feeee but not fe fe{1,3} will match fe, fee, feee but not feeeee.
CHARACTER CLASSES		
[…] **[^…]**	Matches any one of the enclosed characters. Negation – opposite of the above character. It matches any character, which is not included in the enclosed.	[abc] will match either a, b, or c. opposite of the above. NOTE: The negation used within the HTML tags in this thesis e.g. <div[^>]*> indicates that this is the ungreedy expression. It simply matches : <div - a div tag [^>] - don't match an immediate end of div tag *> - match any characters and stops if it matches the end of the tag.
^	Start of a line	^B will match 'B' in Boolean Bool.
$	End of a line	.$ will match s in occurs.
a|b	Matches either 'a' or 'b'.	Organi[s|z]ation will match 'Organisation' or 'Organization'
\s , \d and \w (negated version \S, \D and \W	Matches a white space character such as line feed, a space and a tab, a digit character 0-9 and a word character. \d is equivalent to [0-9] and \w is [A-Za-z0-9]+	\d will match 1 in abc1

While manual construction of regular expression expressions is a widely adopted practical solution, writing precise regular expression expressions to match specified text strings requires experts. Especially when dealing with complex patterns, blocks of simple regular expression need to be combined together, thus generating them manually can be very complicated and lengthy. To avoid these problems, several tools have been developed which automate the generation of regular expression (Barrero et al. 2009; Xhemali 2010a).

Barrero et al. (2009) evolve regular expressions to match the phone number and the URLs that have little variants in the format, which is much simpler to determine compared to the course information; title of course, date, price and location. Lam et al. (2008) suggested that regular expressions can be best applied to static fields and the DOM tree architecture for non-static. According to their definition, static fields are those fields which have fixed format such as an email address where the first part consists of alphanumeric characters, followed by @ symbol, a domain which is made up of characters, a dot and ended with a two or three character word. This thesis is considering this suggestion and several experiments were carried out to prove its effectiveness as reported in Chapter 5.

2.4.4 Evaluation of Information Extraction Systems

To measure the efficiency of the Information Extraction System, MUC has set the evaluation metrics standard; precision rate, recall rate and F-Measures. These metrics determine how accurate the output or the result is to the expected output or how relevant the result is to the problem. For example, if an extraction system returns ten data on which eight are relevant, while it fails to capture twenty of the data, the precision rate is 8/10 and the recall is 8/30. The higher the precision and recall, the better the system performance is. A high precision rate means that there are more relevant results than irrelevant while a high recall rate means that the system managed to retrieve most of the correct results.

The following metrics are defined for this thesis:

precision rate	-	measures the percentage of retrieved instances that were extracted correctly, as represented in equation (e1);
recall rate	-	measures the percentage of actual instances that were extracted correctly, as represented in equation (e2);
F-Measures	-	measures the harmonic mean of precision and recall, as represented in equation (e3).

The metrics (precision and recall) for evaluation for IE was first introduced in 1993 at the MUC-5 (Chinchor & Sundheim 1993) to measure the performance of a system. If the precision is higher, it is likely that the recall is lower (Zheng et al. 2007). This led to the introduction of F-Measures by (Makhoul et al. 1999). These metrics have not only been most widely applied in IE systems but also in Information Retrieval systems.

Table 2.3 Confusion Matrix.

Output	**Predicted**	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

The above has the following meanings in the context of this research:
TP – the extractor correctly extracts the relevant attribute,
TN – the extractor correctly does not extract the irrelevant attribute,
FP – the extractor extracts irrelevant attribute,
FN – the extractor does not extract the relevant attribute, when it should have.

Thus, from the confusion matrix (Table 2.3), the precision rate, recall rate and F-Measure can be calculated using the equations below:

Precision	P = TP / (TP + FP)	(e1)
Recall	R = TP / (TP +FN)	(e2)
F-Measure	FM = 2 * ((P * R) / (P + R))	(e3)

2.4.5 Challenges of WIE

It is well known that all WIE systems attempt to extract all important data and avoid the unwanted ones. In the previous sections, the methods and techniques for WIE were described. This section reviews the general and technical challenges of a WIE from the literature.

Generally, the first main concern for extracting the web content is focused on the quality of the extraction. A robust system tries to avoid extraction of incorrect data and handle imperfect information (missing key data or poorly structured data). Imperfect information is unlikely to provide training data of adequate quality given the more complex and variable language. Missing or null values often caused the system to struggle to recognise the relevance of the information, especially when the rules depend on the location (hierarchical structure) of the value on the web page. This dependency also requires the system to be adaptable to web sources structural change, referred to as flexibility towards changes in Eikvil (1999). In addition, in the case of supervised/semi-supervised method, a major challenge faced is the insufficiency of the training data to achieve high accuracy.

Second is the issue of scaling up the extraction coverage. Early WIE systems were designed to work in a dedicated domain. This posed a problem of confining the extraction task to limited sources and not easily portable. Many researches recently proposed WIE solution for open domain (e.g. Etzioni et al. 2005; Gatterbauer et al. 2007; Banko et al. 2007; Zhu et al. 2009; Wu & Weld 2010; Ji et al. 2013), thus increase scalability. This approach does not emphasise where the information originated.

Approaches relying on supervised (or semi-supervised) learning often require the user to provide numerous training sets to achieve high accuracy of extraction. However, this is time consuming and demands huge human effort. Therefore, when designing the WIE system, there is often a trade-off between the highly accurate performance and highly automated method, i.e., highly automated process (Ferrara et al. 2012).

In many of the WIE studies such as Flejter (2011) and Sarawagi (2008), the cost of processing is another challenge for building an efficient WIE system. Getting relevant information quickly from large volume of data (on the Internet) is especially critical in the field of Business and Competitive Intelligence to enable managers to make informed decisions in relation to critical market conditions. Competitive Intelligence refers to the ability of business organisations to acquire and analyse information from its external environment to support the decision-making process (Chen et al. 2002).

Considering problems of developing a wrapper manually, Machine Learning, Natural Language Processing, Ontology and Genetic Programming are among other methods that have been proposed to automate it. The next section describes these methods in relation to Information Extraction.

2.5 Web Information Extraction Methods

2.5.1 Introduction

In Web Information Extraction, many techniques and algorithms often require customisation especially to cope with different domains having different data types, data patterns, style of presentation and special scripting languages problems. The many methods undertaken by academics are discussed here for an understanding of their application in WIE to overcome these problems in general and the methods applied in this thesis in particular.

2.5.2 Extraction using Machine Learning

Machine Learning (ML) as defined by Mitchell (1997) is *"a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"*. Alpaydin (2010) added that it is also a study of computer algorithms to optimise a performance criterion automatically through experience. It is concerned with approaches to make computer programs

improve by experience, defined by a predefined model or examples rather than by explicitly coded instructions (Guyon & Elisseeff 2003).

Recent studies in the field of WIE are focused on ML. What makes ML popular is the fact that the machine is able to intelligently learn whenever there is a change in the structure, program or data that will improve future extraction. ML can only do the generalisation based on the data that have been seen so far, regardless of the semantics or structure of the data. Given a set of sample data and information about properties of the data, which are its patterns, generalisation allows it to make predictions about other data that it will find in the future and it can continue to learn as new information arrives.

It is important that the program (wrapper) in the ML is able to repair itself whenever there are changes to the web pages such as structure and layout, which will prevent it from extracting data correctly (Lerman et al. 2003). According to IBM (Levesque 2002), the use of ML is motivated by the fact that developing IE systems manually is time consuming and requires linguistics and an artificial intelligence or computational linguistics expert.

ML offers *supervised* and *unsupervised learning* methods. Supervised learning is based on known training input-output pairs to produce a good approximation over these training examples to capture the data. However, this supervised approach depends on a large amount of annotated training data that is often unavailable and requires large effort to create them (Carlson 2010). In unsupervised learning a set of explicit target values is not specified in advance. This aims to find key features from unlabelled examples, called clustering. Another method of ML is the in-between, i.e., semi-supervised. A set of training data from the web page is required to ‘train’ the system. The training uses labelled examples, which may be only positive or a combination with negative examples.

AutoSlog (Riloff 1993), CRYSTAL (Soderland et al. 1995) and Liep (Huffman 1996) were a few ML-based pioneer systems. Autoslog is a domain-specific process and requires annotated training data to analyse the text in order to

produce concept nodes for the extraction rules dictionary. It is able to extract single data at one time. In contrast, CRYSTAL does multi-slot extractions. It requires an expert to manually annotate the data to be extracted. Unlike CRYSTAL, Liep learns dictionaries of extraction patterns from the examples of sentences and events provided by the user. Since then, many other researchers have contributed into this area such as Freitag (1998) and Etzioni et al. (2005). Freitag developed SRV, which is a top-down relational algorithm. It accepts sample texts from the user as tokens and identifies the text fragments using some rules. It then labels these text fragments whether they are for extraction or not. SRV does not require prior knowledge of the format or structure of the text. The KNOWITALL system by Etzioni et al. is an unsupervised system to extract information from the web, where it selects and labels its own training examples using a small set of domain-independent extraction patterns.

2.5.3 Extraction using NLP

Natural Language Processing (NLP) is a subfield of Computational Linguistics and is used to extract information from natural language documents and is suitable for web pages consisting of free text. NLP usually applies techniques such as filtering, part-of-speech tagging, and lexical semantic tagging. These techniques help to recognise the textual content to derive the extraction rules for building relationship between phrases and sentence elements by analysing the syntactic and semantic characteristics of the language. NLP does not depend on any kind of mark-up, thus, it is suitable to be applied to non-HTML documents such as DOC and PDF. However, the problem of identifying the correct sense of a particular word in a particular context still arises in NLP and the ability to emulate human understanding of natural text is still a long way away.

There are various researchers working on NLP, for example, Andersen et al. (1992) has introduced JASPER, which is a commercially used IE system and Cunningham et al. (2002) and Morton (2000) proposed GATE architecture, which can be used in IE applications. Some other researchers went on

combining NLP with other techniques to improve the extraction rates and applications to diverse domains, such as NLP with ML, commonly used in Bioinformatics and medicine (Alphonse et al. 2004; Buyko et al. 2006; Sokolova et al. 2006; Lussier et al. 2006), NLP with Ontology such as ontology lexicons (Cimiano et al. 2007), a question answering system (Vargas-Vera & Motta 2004) and digital information retrieval (Jeschke et al. 2007).

2.5.4 Extraction using Ontology

Ontologies are defined as “content theories about the sorts of objects, properties of objects, and relations between objects that are possible in a specified domain of knowledge” (Chandrasekaran et al. 1999). Ontologies have been used to extract information from diverse domains such as literature (Muller et al. 2004; Milward et al. 2005), tourism information (Maedche et al. 2003), soccer matches (Buitelaar et al. 2006) and multimedia content (Paliouras et al. 2011). An ontology represents the vocabulary, which describes the concepts of a particular object, rather than the object itself, that is intended to be captured and it is often specific to some domains. For WIE, this requires careful analysis of the specific information/entity and its relations to other information that can exist within the domain. The ontology approach aims to solve information extraction by defining the relationship, lexical appearance and context keywords of the data of interest in a document. It relies directly on the data as opposed to other methods, which rely on the structure of data presented within the document for the generation of rules or patterns for the extraction.

Ontology supports knowledge sharing and reusability but defining a common ontology or a common standard for a particular entity, which can be applied across various domains is still lacking. This means ontology requires a large knowledge base in order to capture the various forms of presentation of a specific piece of information.

Although ontologies can be used to model data and define its semantics before the extraction process, the quality of the ontologies is important, which will determine the success of a WIE system based on ontology (Ontology-Based Information Extraction - OBIE) (Wimalasuriya & Dou 2010; Buitelaar et al. 2005). Maintaining the ontologies is time consuming (Labský et al. 2008), especially if it is done manually because an information needs to be analysed and the expressions need to be evaluated, such as to define the meaning of a particular word or its context in a particular domain. For example, in a case where new products are added or words with different semantics are used in a web page, the ontology needs to be updated for the extraction to work properly.

According to Brank et al. (2005) evaluation of the resulting ontology is an extremely challenging task. This technique only provides meaningful results if the data are of good quality (Juffinger et al. 2007) and if it fails to identify the domain-specific concept of the data; it then goes back to the human to refine this ontology constructor.

2.5.5 Extraction using Genetic Programming

Genetic Programming (GP) aspires to do the same as Machine Learning, but GP is to induce a population of computer programs that can improve automatically as they experience the data on which they are trained. Although GP has been actively used in other areas such as games and bioinformatics, it has seen little application in WIE. Only recently, the idea of using GP in WIE has proved to be useful and attracts a considerable interest to evolve the extraction rules to discover the information.

One approach to automate the extraction task is to evolve a regular expression. For so many years, regular expression has been used as a matching tool in practical WIEs. However, this is a very complex task and a huge manual effort is needed when composing a high quality and a complex regular expression for the WIE tasks. Latter innovation examines ways for automating this composition, in particular, through application of evolution

process. Few researchers (Bartoli et al. 2012; Xhemali 2010a; Barrero et al. 2009; Li et al. 2008) have successfully applied automation to evolve regular expressions, which has resulted in better extraction success rates. However, the grammatical rules to support the generation of the regular expressions are manually created. Thus, maintaining these rules demands a significant amount of expert effort. Inspired by this achievement, this thesis investigates how to further improve the extraction task towards automating the rules refinement/addition to keep up with new data representation, by being given new examples for training.

Challenges in WIE using GP

Developing WIE incorporating GP principles poses the same general challenges as stated in this chapter, with a few additional ones as below:

1. GPs can get stuck in local optima regions of the search space
2. Operating on dynamic data set is difficult as it is likely that the GP will converge early on towards the earlier solutions which may not be valid for the later data.
3. The extraction patterns created may not be able to extract information properly and this would require a repairing mechanism to extract only the valid ones.

The next section introduces an important method used in this thesis; the principles of Genetic Programming, which is widely applied in evolving computer programs.

2.6 Evolving Computer Programs

2.6.1 Introduction

The pioneering work of generating programs using a Genetic Algorithm was by Cramer (1985). His work was to generate a system, which accepts two inputs and produces a single output multiplication function. Today, in this

area, similar work has been done, taking on the evolution of human biological genetic processes such as Genetic Programming (Koza 1992) and Grammatical Evolution (Kuroda et al. 2010; Hugosson et al. 2007; O'Neill & Ryan 2003, 2001).

Genetic Programming is one of the most prominent computational techniques for evolution, branching from Genetic Algorithms, which was popularised by John Koza (1992) along with Genetic Algorithms (Holland 1975). In GP, solutions to problems are represented as computer programs. As defined by Koza, GP is "*a domain-independent problem-solving approach in which computer programs are evolved to solve, or approximately solve, problems.*" His approach started with a tree representation of a program in order to generate variations of solutions. A more general definition by Banzhaf (1994) is *"the direct evolution of programs or algorithms for the purpose of inductive learning"* and (Poli et al. 2008) describe it as *"At the most abstract level GP is a systematic, domain-independent method for getting computers to automatically solve problems starting from a high-level statement of what needs to be done."* GP has been proven successful to solve *weak* problems and does not require explicit task knowledge.

2.6.2 Genetic Programming Terminology

The following terms in Table 2.4 are used in the rest of this thesis, when referring to GP.

Table 2.4 Meaning of terms used in this thesis.

TERM	DEFINITION
Genome	an individual encoded set of parameters, which defines the proposed solution.
Genotype	a unique encoded set of parameters which may represent several Genomes
Individual	a member of the population consisting of a genome and its fitness level.
Genetic Operators	the transformation operators which act on a single genome or two genomes. Primary transformation operators are crossover and mutation.
Fitness Function	a measurement of the quality of the individual, i.e., how close an individual is to the actual solution
Phenome	an individual representation of the solution and is the transformation of the Genome
Phenotype	a unique representation of the solution which may represent several Phenomes
Search Space	the set of all programs/elements determinant by the given programming language that could be found by the GP algorithm

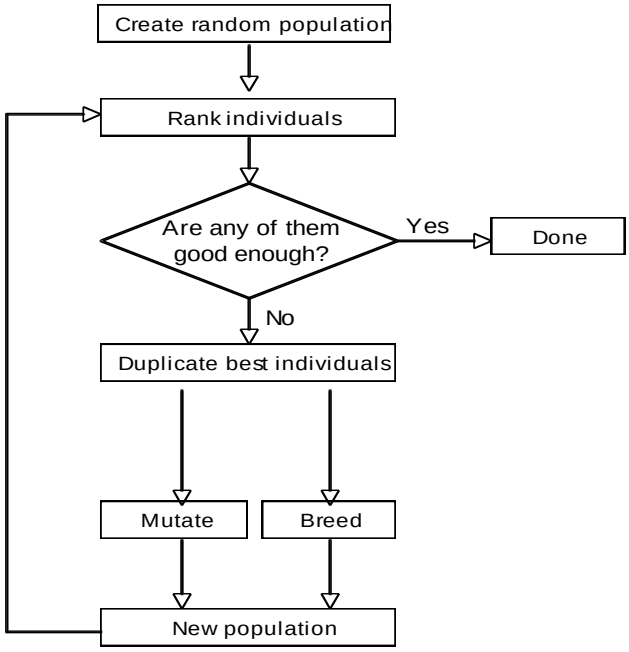
2.6.3 Basic Concepts of Genetic Programming

According to (Poli et al. 2008), Genetic Programming, abbreviated as GP, comprises several general steps (Figure 2.7). First, individuals are presented in the initial population. The initial population can be created by different methods;

(1) **random** – this is the most unbiased method. Generation is using pseudo-random number generator and it can provide great variation between individuals,

(2) **seeded** – this method biased the population with known solutions to the problem to start with. Although this can make it harder for the GP to find better solutions, seeding the random number generator helps to make the run repeatable.

(A)



(B)

1: Randomly create an initial population of programs from the available primitives
2: Repeat
3: Execute each program and ascertain its fitness.
4: Select one or two program(s) from the population with a probability based on fitness to participate in genetic operations
5: Create new individual program(s) by applying genetic operations with specified probabilities
6: until an acceptable solution is found or some other stopping condition is met (e.g., reaching a maximum number of generations).
7: return the best-so-far individual.

Figure 2.7 (A) The basic GP flow diagram (B) The basic GP Algorithm (source: Poli et al. 2008)

A genotype to phenotype mapping is performed to decide how good an individual genome is compared to other individuals in the population; the phenome is evaluated rather than the genome. A genotype (genetic codes for a solution) refers to a specific unique sequence of genes. A gene is an integer value, which represents a terminal or a function in the phenotype. A phenotype refers to the solution, which corresponds to the genotype. Basically, a genotype acts as a guideline for building a phenotype, which is strictly dependent on the value of each gene. The genotypes are made up of either a variable length or a fixed length of genes. The size of the population

influences the result of GP and can vary from one problem to another. The smaller the population, the less time it will take to calculate the fitness of each generation. However, smaller populations reduce the chances of generating genotypes with good characteristics in the initial generation. Koza (1992) suggests that fitness evaluation is done by comparing the required output with the actual output produced by the algorithm.

Secondly, the breeding process is repeated until one of the termination conditions is met. There are three ways to terminate a GP; specify the maximum number of generations or until the perfect solution is found (a specific fitness score is reached) or the population has not improved for a predefined number of generations.

In the third step, those genomes are transformed into phenomes and the fitness of each individual is measured. The goal of a fitness function is to guide the evolutionary process through the problem space to arrive at an optimal solution (Wilkerson & Tauridtz 2010). The fitness score is calculated on individuals based on how close the output is to the expected solution. This measurement is used to determine the quality of individuals in the population whether it has a higher probability for reproduction in the next generation or being discarded. The fitness function falls into two categories; single fitness and multi-objective fitness. Single fitness is a conventional method, which concerns satisfying a single objective. Multi-objective concerns the decision-making process involving two or more objective functions aiming at achieving optimisation to solve a problem. This commonly involves trade-offs between all objectives. Weights can be assigned to each objective to define their importance or dominance. Other multi-objective systems, such as NSGA II, use the concept of non-dominance to rank the phenomes (Deb 2002). A *phenome A* dominates another *phenome B* if all objectives in A are preferable to those in B. B is not dominated by A if at least one measure of B is better than the corresponding one in A. This avoids commitment to a particular ranking of objectives and delivers a Pareto Optimal front in the solution space.

Fourthly, GP selects pairs of better individuals from the parent population, which will be used to create the next generation of individuals, with a hope that their offspring will have even higher fitness. There are several selection techniques introduced by researchers in this field but the most popular ones are the Tournament selection (Brindle 1981), Fitness-proportionate selection (De Jong 1975) and Rank selection (Baker 1985).

Tournament selection: This method selects a group of individuals randomly from the population and runs several 'tournaments' or competitions to get the individual with the highest fitness among the other members. The number of individuals in a group depends on the size set for this tournament, called the tournament size, ranging from two individuals to the number of individuals in the population. If the tournament size is too small, it is possible to have some individuals not selected at all. Barrero et al. (2009) in their experiments demonstrated that the tournament size has a significant impact on obtaining a faster convergence while avoiding local maxima.

Fitness-proportionate selection: This method is also known as roulette wheel selection. For this method, individuals are also selected randomly from the population and assigned a slice of the wheel with the size proportional to individual's fitness value. This means an individual's chance of being selected is proportional to its fitness value. Similar to the roulette wheel in a casino, this method spins the wheel to get the reproduction candidate.

Rank selection: This method ranks individuals in the population and each is assigned a numerical fitness value according to their rank, i.e., rank 1 to worst individual and rank *n* to the best, where *n* is the number of individuals in the population. Like Fitness-proportionate, each individual's probability of selection is proportional to their rank.

Despite giving a better chance to higher fitness individuals, tournament also gives a chance to the less fit genomes to be included in the

tournament for selection. By contrast, in both Fitness-proportionate selection and Rank selection, individuals with higher fitness are more likely to be selected for the reproduction. The Fitness-proportionate selection could lead to a problem of premature convergence (causes the search to narrow down too quickly) when the fittest individuals dominate, whereas in Rank selection, there could be a problem of slow convergence. Detailed information on these methods is given in the literature, such as Blickle and Thiele (1996) and Ma (1995).

The fifth step is also known as the 'reproduction' process. Reproduction creates the next generation of solutions (offspring), which ideally share many of the useful characteristics from their parents (in the current population). The survival of any individuals in the current population depends on their fitness. A better fitness individual will be allowed to survive by copying it into the new population; otherwise it will be replaced by the fitter offspring. The two main genetic operators are crossover and mutation. There are several possible methods for both crossover and mutation. The simplest crossover method is single point crossover, followed by multi point crossover; however uniform crossover has several advantages. It is efficient and simple to program.

Uniform Crossover involves the genes of two parent individuals being combined to produce a new individual. This is done by selecting a gene from a genome with a certain probability, usually 50% (Sywerda 1989; Jones & Hinde 2007). This means, each gene of the parent individuals has the potential to be included in the offspring, and so there is no identifiable crossover point. For each gene, a random number (real number) is created, which is referred to as the mask and only if this random number is less than or equal to the probability, then the gene in the first parent is used rather than the gene in second parent to produce the first offspring. The second offspring is created from the opposite, that is, the remaining gene.

Mutation of each offspring takes place following the crossover operation. It is simply making a small random change in the genome (in practice,

$1/n$ is the common mutation rate where n is the number of genes in the genome) to explore new possibilities in the search space. The number of genes in a single individual being mutated depends on the mutation rate specified. Table 2.5, 2.6 and 2.7 illustrates the GP operators applied to the selected individuals, where the random number is picked from a range of numbers between 1 and 225.

Table 2.5. Example of parents selected from parent population

Parent1 :	25	125	120	53	3	20	
Parent2 :	24	158	36	21	124	5	7

Table 2.6. Application of a Uniform crossover operator to the offspring.

Random number :	0.2	0.6	0.3	0.65	0.55	0.25	
Offspring1 :	25	158	120	21	124	20	
Offspring2 :	24	125	36	53	3	5	7

Table 2.7. Application of a Mutation operator after the crossover.

Random number :	0.2	0.01	0.3	0.08	0.55	0.25	
Offspring1 :	25	18	120	212	124	20	
Random number :	0.06	0.11	0.14	0.25	0.5	0.92	0.66
Offspring2 :	111	125	8	153	3	144	7

GP has been used to solve problems in various fields, such as, medical (Guo & Nandi 2006; Hong & Cho 2004), Railway platform allocation (Clarke et al. 2010), robotics (Konig & Schmeck 2009), programs (Koza 1992,1994; Withall et al. 2009; Xhemali et al. 2010b), symbolic regression (Castillo et al. 2005; Smits et al. 2006) and information extraction (Xhemali 2010a; Barrero et al. 2009). Before the GP system begins, Walker (2001) specified several control parameters that need to be decided, which are:

1. Population size. The larger size of population helps to increase the chance of evolving a solution as it allows for greater exploration of the problem space at each generation.

2. Maximum number of generations. This parameter is to control the run, which aims to provide the evolutionary program ample time to evolve a solution or approximate solution to a problem. Although

greater maximum number has a higher chance for this program to produce a solution. However it is not guaranteed as in some cases individuals fail to show any further improvements, which means this evolutionary program needs to restart with a different initial population.

3. Probability of crossover. This parameter determines the probability of an individual to undergo crossover before the decision, whether to move it to the next generation or eliminate it.

4. Probability of reproduction. This parameter is used to define the proportion of individuals in the population to be reproduced.

However, there is no guarantee of success. The success of GP depends on careful selection of the control parameters (Poli & McPhee 2009). According to Poli and Langdon (2007), GP search space is extremely large and only a tiny fraction of it can be examined by any search algorithm.

There has been a large amount of research on problem modularisation and its effect on scalability. Four of the most popular extensions of genetic programming are Automatically Defined Functions (Koza 1994), Cartesian GP (Miller & Thomson 2000), Genetic Network Programming (Katagiri et al. 2000), and Dynamically Defined Function (Hemberg et al. 2009). In practice, modularisation can be seen as separating partial solutions into independent modules that each solves one aspect of the sub-problem. In evolutionary algorithms, the same concept has been researched and successfully applied. These modularisation methods are described in more detail in the next section.

2.7 Variation of Genetic Programming

In this section, the most relevant works on function evolution are briefly reviewed, where researchers reported an increase in performance compared to the standard GP. All these works are based on a top-down approach, where functions are derived from the main program. In the context of this thesis, a novel bottom-up approach is introduced where useful functions are evolved separately from the main and this newly evolved function definition is added to the core grammar for future use. A script was developed to automatically add the newly evolved function that may be used to solve a different computer problem.

The idea of function evolution was introduced by John Koza when he expanded his earlier work to apply the modularisation concept to the generated program (called Automatically Defined Functions); see (Koza 1994) for more details. This allows for the automatic creation of parameterised functions that can be invoked from the main program while the GP is concurrently being evolved. GP with ADF (ADF-GP) automatically decomposes a program into a set of modularised subprograms during runtime, with each solving a sub problem with the capability for reuse and then reassembling to sol**v**e the original overall problem. ADF-GP manipulates the program tree and the functions are created from the sub-trees. In standard GP, these sub-trees (partial solutions) are prone to modification by genetic operators. As a result, it would become more difficult for the evolutionary process to find a useful solution, leading to an increase of computational effort. Thus, ADF-GP introduces compression and decompression techniques to protect these sub-trees.

This approach allows for generation of larger and more complex programs and has a benefit of significant reduction in the computational effort compared to GP without ADF. However, before evolution commences, the number of functions and their parameters need to be defined initially, although during runtime, these parameters are allowed to change with no human intervention. The functions are evolved to strictly define movement (left, right, top and bottom).

Since his seminal work, there has been a remarkable amount of work to establish a theory of modularisation, to find effective and efficient methods for optimising the evolution solution, and to apply those methods to practical problems. A variation of ADF-GP can be seen in the work of Harper and Blair (2006) through their paper entitled “Dynamically Defined Functions in Grammatical Evolution”. In Harper’s Dynamically Defined Functions (DDF), functions are dynamically created using a core grammar represented in BNF notation as in Grammatical Evolution (GE). GE is an extension of GP to evolve programming codes to solve a defined problem; its strength lies in the utilisation of grammar for any chosen language, as long as it can be expressed in BNF notation. This method uses a linear, variable length genotype made up of a string of 8-bit binary numbers.

The functions created by DDF are then automatically appended into this grammar. Contrasting ADF, Dynamically Defined Functions (DDF) does not require the user to specify the number of functions and their parameters prior to evolution. The functions, which may have any number of parameters, can be invoked by the main program, independent of any special-purpose operators or constraints. DDF has proven particularly successfully in the MineSweeper problem (Harper & Blair 2006).

Because there is no specific size of individuals being set, this method is facing a danger of insufficient integers to complete a code and a program “bloat”. A program “bloat” is when the evolving program has a rapid increase in size over some generations. This individual is either discarded or a specific method is then put in place to overcome these issues. The functions in DDF

are not evaluated and are not defined to do a specific task. One of the weaknesses of this approach is that because it does not require user involvement to pre-define an optimal architecture, it is likely to experience a large search space to be able to get to the solution quickly compared to ADF.

The first work to introduce CGP was published by Miller in 1999 in a paper "An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach". CGP was originally concerned with providing an effective method for evolving digital electronic circuits (Miller et al. 1997). In the latter development, CGP is applied to computer programming. Using this method, the computer programs are encoded in the form of a linear string of integers representing an indexed graph. There is no restriction as to how many inputs the program can take and how many outputs it is allowed to produce. However, each node must have a function and a set of inputs. An input is either the output from the previous node or an initial program input.

Like standard GP, genotype-phenotype mapping is also very important in CGP. The genome represents some functions and node connections, producing an executable program (phenome). Although the genomes are fixed length, the phenome length varies. This is because the nodes, which are encoded by a number of integers (genes), are not required to be connected to each other. Any unconnected nodes will not be processed and they do not have any effect to the program's behaviour. Unlike Banzhaf (1994), no repair is necessary in CGP due to application of certain restrictions on the gene associated with the input, output or function that control the validity of the output.

CGP is reported to outperform GP with ADF over some kinds of problems of sufficient difficulties (Miller 1999; Miller & Thomson 2000). However, node outputs (Automatic Re-used Outputs – AROs) can only be re-used if they have the same inputs. Other researchers, (Walker & Miller 2008) present an extension to CGP called Embedded CGP (ECGP). In this new method, a similar form of ADF was implemented to allow for construction of modules

(composition of primitive rules), which can be automatically called and evolved. One of the drawbacks of this approach is scalability. This is because the module has size limitation, which restricts the maximum size of the genome.

2.8 Critique

Dawson (2009) states the importance of identifying the research gaps through the critical evaluation of the literature in the field to ensure work originality and that unnecessary duplication is avoided. Based on the review of the literature in this Chapter, with regards to the wrapper generation and the quality of the extracted information, it therefore can be concluded that there are indeed a number of research gaps that warrant experimental exploration. The research gaps identified below form the fundamental foundation for the progression of the work in this thesis, as well as outlining the specific area of knowledge that this thesis seeks to contribute.

Wrappers to extract information required by the user from the web sources are well researched. They work well with HTML web pages, which assume that some information on their structure is available. However, it is well known that wrapper generation and maintenance is difficult (Laender et al. 2002; Ferrara et al. 2012), which requires human experts in this area. It appears that there is a lack of empirical evidence attempted to explore the evolution of wrappers (represented in regular expressions) using evolutionary algorithms such as Genetic Programming, in Web Information Extraction systems, in particular, the increment of extraction rules through semi-automation. The intention is to allow for the generation of the extraction patterns that can be applied to the 'never seen before' web pages. Regular expressions are a well-established tool in a variety of application domains, particularly in text processing (pattern matching), and continue to be the most extensive practical applications because of their flexibility and expressiveness.

To the best of researcher's knowledge, there are only three researchers that have attempted to automate the evolution of regular expressions for WIE task; Barrero et al. (2009), Xhemali (2010a) and Bartoli et al. (2012). On one hand, Barrero et al. research concerned the extraction of URLs and phone numbers from web sources using a multiagent system (MAS). MAS has agents to manage a population of fixed length chromosomes. Part of one population may migrate from one agent to another during evolution process, which eventually forms variable length genomes from which the basic regular expressions are created. Another agent then integrates two or more regular expressions using a subset of regular expression operators (e.g. |, (,), + and ?) forming rules like X|Y, X+Y?. The next stage is to filter these rules to select the composition, which scores better on a validation set. The drawback of this technique is that it favours the X|Y combination, making the last stage useless. For a discussion of Xhemali's work, refer to Chapter 3.

Bartoli et al. (2012), on the other hand, focus on the extraction of phone numbers and HTML titles. This differs from the other two approaches above as it is a semi-supervised method. The user does not need to possess any technical knowledge, other than providing a set of labelled examples (a pair of strings indicates a positive example and just one string indicates negativity). This thesis follows a similar path, which is to optimise the evolution of regular expressions for extraction of data from web sources. Unlike Bartoli, the regular expression not only represents the format of the string but also combines it with DOM tree representation. Section 5.3 discusses this approach in relation to the approach proposed in this research.

Review of Methods for Application

The fundamental concept of standard GP was discussed in this chapter. Some interesting work and relevant variants of GP extended version such as ADF, DDF and CGP are also described. The breaking down of program into reusable subprograms has been the main focus in these extended methods, which allows for the generation of a complex and larger program. The idea of using subprograms has influenced the technique presented in this thesis to

improve the performance of a sufficiently complex problem as detailed in Chapter 4.

Although GP has been in widespread use since 1972 to tackle many areas including gaming, bioinformatics, robotics and timetabling, it has only recently made its mark in Information Extraction and an example of work applying such a method is introduced in Chapter 3. GP helps to find a solution through a repetitive process by improving on the available or learned solution. Although EC approach (GA or GP) has only been recognised to be useful for WIE compared to other approaches, researchers, such as Gonzalez et. al. (2010), Xhemali (2010a) and Barrero et. al. (2009) have produced successful application to evolve regular expressions automatically through the use of grammatical rules in producing matched data patterns.

In many cases, the development cost for matching new or unknown data format to the WIE system remains substantial and results are not directly reusable for other problems. WIE systems lack knowledge to be sufficiently flexible to take advantage of repetitive patterns in that domain. To ease this difficulty, human intervention is the main motivation applied in this thesis to recognise the commonality. With the new approach presented in this thesis, the emphasis now is shifted from a hard and expertise-specific task of building and rebuilding the hand-crafted extraction rules to a lighter and a more general task of providing new training data for the system to learn.

NLP is not considered for this research, as the research does not involve analysing and extracting multiple sentences where the grammatical syntax can be observed in the sentences constructed as in the case of extracting email contents in (Tedmori & Jackson 2009). This research only aims to extract specific pieces of information from the training courses web page, which are the title, location, date and price. Furthermore, because the training course web page presents information not only in a form of grammatical text and paragraph which suits NLP, but also the majority are presented in tables and lists which NLP cannot handle, therefore NLP approach is ineffective (Lam et al. 2008). It can be concluded that this technique is not feasible to be

investigated further.

Ontology approach is also not considered for this research. Because this approach requires clear definition of the relationships between entities (data value) and most ontologies are created to be domain specific, in the context of this research, this would be notably difficult, especially for ATM as ATM has no experts in this area and maintaining a huge knowledge base would be expensive for them. Moreover, because this TS-WIE is an extension of the automatic WIE, which was developed using GP, and does not incorporate ontology, major reconstruction of the automatic WIE system would be needed.

This research concerns the semi-supervised learning for wrapper generation. The wrapper is generated automatically based on the positive training examples annotated by a human expert and the algorithm uses this wrapper to guess the instances of course attributes[2] from a given web page. In the light of the studies cited here in this chapter, the question remains, can structural analysis and lexical analysis, with learned rules from minimal positive examples improve the accuracy and scalability of the WIE system?

2.9 Chapter Summary

Sarawagi (2008) highlighted that designing and implementing an effective IE system poses some design challenges such as accuracy and efficiency. This is because information is represented in a variety of transformation and structural differences. This chapter has reviewed many advances in the WIE methods, ranging from manual approach to automation.

Wrappers have been used to recognise the information of interest on the web page. In the early days, wrapper was handcrafted but this is too human expensive and it was domain dependent. To overcome this problem, later approaches introduced automation and semi-automation. However, the

[2] In this thesis a "course attribute" will mean a title, location, cost or duration of a training course and where there is no likelihood of confusion with other attributes (like HTML attributes) then "course attribute" may be abbreviated to just "attribute".

current IE researches have shown that automation is a complex task and achieving high quality extracted information is still an on-going problem.

This chapter has attempted to provide a detailed overview of the existing literature relating to the area of Web Information Extraction (in particular semi-automatic approach) and Genetic Programming (GP). Semi-automation is considered a main subject of wrapper generation, and wrappers are important in the extraction process to be investigated in relation to human effort from both a theoretical and practical point of view. Not only did this chapter provide insights on the discovery of important research gaps within the study, it also offered justifications of the potential of GP towards enhanced methods of WIE solution. The following chapter discusses the detail of the two related works, which provides a fundamental foundation for the progression of the work in this thesis.

Chapter 3

Use of Genetic Programming to Evolve Patterns

3.1 Chapter Overview

Chapter 2 discussed the literature of the Web Information Extraction in general. This chapter introduces two strands of work by two other researchers that formed the starting point for this work. For each of the two strands of work there are two sections. The first section briefly describes the work and its features. The second offers a critique and, in particular, identifies significant weaknesses of the work, which will be addressed by the work in this thesis. Specifically, this chapter discusses how these works can be used as a foundation for developing a suitable WIE software tool and which area of the automatic WIE is a good foundation for an improved WIE solution.

The chapter closes with a summary of the lessons learned from the previous work and Chapter 4 and Chapter 5 report the features that will be incorporated in the new work.

3.2 The Evolution of Complete Software Systems

3.2.1 Introduction

This section introduces two researches on software system evolution; Withall (2003) and Xhemali (2010b). Withall started an evolution of software, which examines finding a solution for the 'sorting program' using fixed-length genotypes. Sorting is best performed using iteration or recursion and evolving the algorithm is a significantly complex task (Kinnear 1993). Xhemali later extended Withall's work by introducing variable-length genotypes and XML-based rules. Both works used Genetic Programming (GP) to evolve the 'sorting' program through manipulation of hand coded and rigid rules.

3.2.2 “Sorting program” evolution - The work of Withall

Withall et al.(2009) evolved a sorting program using a reduced programming language subset, which is coded in PERL. This program evolution uses linear representation. They propose the fixed-length blocks genotype. The phenotype is produced by individuals in a population (genomes) consisting of 40 genes each. The genomes are represented as a string of integers, where each integer represents a different gene. Each genome is divided into blocks of four integers and each block produces a single statement in the resulting phenotype. This means the phenotype has ten statements.

The fixed-length blocks are padded with redundant genes called ‘padding gene’ to avoid the problem of insufficient genes in the variable length genotype. Withall argues that the padding is useful to maintain the same block lengths and to preserve characteristics of parents that can be inherited by the offspring to ensure efficiency during the crossover, mutation and mapping processes. This aims to minimise the characteristic gap between parent and the offspring caused by a single mutation. Therefore, in a case where a particular program structure or statement requires fewer genes, the unused genes in that block will be ignored. This should ensure that the next statement/structure translation would start from the first gene in the block.

The generation of the ‘sorting’ program is assisted by the concise programming language subset, which is coded in PERL, describing the rules to form a particular statement or structure (see Figure 3.1 for an extract of the rules). The Genotype-Phenotype mapping is applied to transform a string of integers, which make up a genotype used for genetic manipulation, to a sorting program in PERL for fitness evaluation. To avoid problems such as infinite loops, a restrictive approach was introduced, i.e., allowing only limited time for each execution.

```
# ----
# for
# ----
sub oyster_for {
  my ($prgm, $v1, $v2) = @_;

  # Decode genes
  $v1 = $counter[$v1%($#counter+1)];
  $v2 = $lsize[$v2%($#lsize+1)];

  # Generate Code
  if(1)   {
    $prgm .= "for $v1 (0..$v2){\n  \$runtime++;\n  die if(\$runtime > \$timeout);\n";
    $ob++;
  }
  return $prgm;
}

# ----
# double
# ----
sub oyster_double {
  my ($prgm, $v1, $v2, $v3) = @_;

  # Decode genes
  $v1 = $counter[$v1%($#counter+1)];
  $v2 = $lsize[$v2%($#lsize+1)];
  $v3 = $counter[$v3%($#counter+1)];

  # Generate Code
  if(1)   {
    $prgm .= "for $v1 (0..$v2){\n  for $v3 ($v1+1..$v2){\n  \$runtime++;\n
    die if(\$runtime > \$timeout);\n";
    $ob+=2;
  }

  return $prgm;
}
```

Figure 3.1 An extract of rules stored in a PERL file to assist the generation of 'sorting' program

First the genome is separated into blocks of four genes. A gene is used to determine the rule to follow by using modulo operator (written % in the code). This manipulation of genes using modulo parses the statements and arguments of every block. Because the first gene of each block specifies the type of *statement*, for example, a "for statement", "if statement", "assignment statement" etc., by having the gene value modulo the size of the statement type, would map to one of the candidates. For example, if the gene value is 200 and there are five *statement* types, then 200 Mod 5 = 0 and so would pick

the first *statement* type from the *statement* options. The remaining genes will be processed in similar ways depending on the requirement of the rule (see Figure 3.2 for an example of the mapping process). For this approach, a rather higher mutation rate of 0.1 is used to get good results. Although this method guarantees consistent mapping and a complete program is generated, a 'repair function' was introduced to ensure that the 'sorting' program produced follows the correct syntax. The same correcting mechanism concept was observed in the earlier work of Banzhaf (1994).

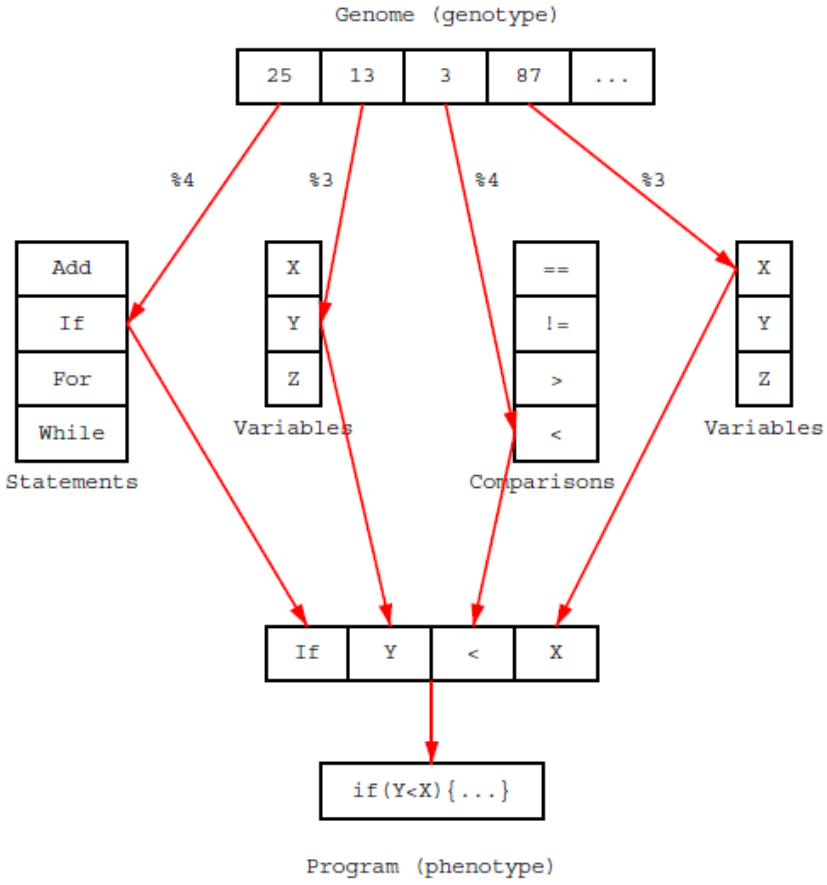


Figure 3.2 An illustration of the formation of a phenome from a genome (Source Withall, 2003)

A similar work was later carried out by Xhemali (2010b) but with variable-length genotypes instead of fixed-length. Here they introduced XML rules of the programming subsets syntax, which guides the mapping of the genotype into a valid phenotype. The main contribution from this was to remove the translation process from a hard coded system to a table driven approach, which could then be modified and extended by an external process.

3.2.3 “Sorting program” evolution - The work of Xhemali

A new approach to efficiently evolve a sorting program and also an extension to Withall’s work above was introduced by Xhemali (2010b) in a paper entitled “Genetic evolution of sorting programs through a novel genotype-phenotype mapping”. In their paper, they outlined three main differences distinguishing both works. Firstly, instead of using fixed-length genotypes, this work is based on variable-length genotypes. Secondly, XML rules (Figure 3.3 shows the structural content of the XML file), which are stored in a file external to the GP program, have been applied to achieve the mapping of genotype to phenotype. Thirdly, the ‘sorting’ program was evolved in VB.NET 2008 whereas Withall’s target language was PERL. This demonstrates the advantage of these works as being language independent.

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
 <rules>
  <!-- IF -->
   <rule id=”0” start="IF" end=”END IF” nested="true">
    <component id=”0”>1</component>
    <component id=”1”>2</component>
    <component id=”2”>3</component>
   </rule>
  <!-- FOR -->
   <rule id=”1” start="FOR" end=”NEXT” nested="true">
    <component id=”0”>1</component>
    <component id=”1”>4</component>
    <component id=”2”>1</component>
   </rule>
..
</root>
```

Figure 3.3 Rules stored in a XML file to assist the generation of a ‘sorting’ program (source Xhemali et al. 2010b)

The process of mapping the genotypes to phenotypes is assisted by the modulo operator, which is the same method used by Withall. Xhemali also introduces a ‘repair function’ that deals with fixing any syntactically incorrect program structures produced by the evolutionary system, such as adding an ‘ENDIF’ statement at the end of an IF statement.

Similar to the method introduced by Ryan, Collins and O'Neill (1998) in their paper "Grammatical Evolution: Evolving Programs for an Arbitrary Language", this method poses a disadvantage of characteristics inheritance. This means an earlier change in the gene value of a genome (through crossover or mutation) can change the entire construct or type of statement following, which results in the child having little similarity to its parents.

3.2.4 Discussion

Xhemali's work was inspired by Withall's to evolve a complete program proposing different parameters and breeding techniques. Table 3.1 shows a comparison between the parameter values and the GP operators applied in both works.

In Withall's work, the strength lies in preserving the characteristics inheritance between the parents and the offspring by introducing a fixed-block genotype to correspond to a single code line in the phenotype and also avoiding insufficient genes required to produce a valid phenome. He states that the offspring should inherit good characteristics from the parents and points out that the effect caused by the GP operators to the offspring should be minimal. Xhemali, on one hand, did not see this as the main obstacle to introduce a variable length genotype to produce a valid phenome. On the other hand, she introduces a fixing method to ensure that the phenome is syntactically correct. Furthermore, if a particular genome encodes to an incomplete program, Ryan et al. (1998) and Paterson and Livesey (1996) resort to gene reuse or randomly extend the genome and similar approach is also implemented in Xhemali's work.

On the contrary, Xhemali focused on moving the programming language subsets (rules) into an independent and hierarchically structured file (XML file). This provides an advantage of presenting the rules in any programming language and it is also easier to extend.

Table 3.1 GP methods - Withall versus Xhemali.

Parameter	Withall	Xhemali
Genotype length	Fixed-block	Variable
Genotype representation	String of Integers	String of Integers
Selection	Simple Fitness proportionate	Tournament
Crossover	Uniform	Uniform
Mutation probability	10% of the 40 gene-length genotype	One in each genotype
Language subset (grammar)	Rigidly coded in PERL and stored in the source code.	Rigidly coded in an XML file
Repairing function	Yes	Yes
Population size	7 genomes	10 genomes
Phenotype size	Fixed	Variable

3.2.5 Critique

Based on the analysis on the techniques used in the previous works, there are four issues found, which are:

- In both works, the grammar was not properly constructed. Some shortcuts were introduced which makes the rules very rigid and difficult to expand. In Withall's solution, the rules are coded using PERL, which can only be maintained by PERL programmers. Whereas, Xhemali's XML file uses rules in terms of number reference which is quite difficult to read and follow.

- The grammar rules built were dependent on the controlling function in the evolutionary program, which makes it difficult to apply any changes to the rules. This means if the rule component has to be altered, the changes need to be reflected in the program and this requires the expertise of a programmer. This means that the intention to develop a GP system that could be extended by an external process without modifying the main program code was not fulfilled.

- The fitness test function used (both works use the same test), which is responsible for measuring the fitness of a particular genotype to be either carried forward to the reproduction process for the next

generation or be discarded from the population, produced invalid measurements. Note that the fitness is measured based on the output produced by the evolutionary program. The fitness function was derived from the formal specification for the program with every conjunctive requirement that was met led to an increment of 1.0 in the fitness. Although this is a valid measure, it was found that the longer the output, the higher the fitness score it would get. This is misleading because if the output is longer than the expected result, it should be given a much lower fitness score. Clearly deriving the fitness function from the specification is a useful idea, but the implementation falls short of an ideal measure.

- Although both works use a 'Repairing' function to ensure that the generated 'sorting program' is syntactically correct, it would be difficult if a new structure or a new statement rule is added to the grammar, which requires a different fixing solution.

To solve the above issues, this thesis proposes a new GP approach. During the initial stage, the effect of introducing modularisation and generic programming language subsets to the performance of the GP was investigated. The subsets were applied in the transformation process in the GP method. The purpose is to evolve a complete software system ranging from solving simple problems such as addition of integers, to increasingly complex problems such as sorting integers in descending order. Following this idea, several experiments were performed to study the effect of this new approach on the evolutionary process; in terms of the time taken to find a solution and the fitness evaluation requirement (Chapter 4 has the details of this approach).

The next section describes Xhemali's automatic Web Information Retrieval/Web Information Extraction system, which provides a motivation for an improved WIE solution.

3.3 Automatic WIR/WIE System - Xhemali

3.3.1 Introduction

This section introduces a separate piece of work by Xhemali called an "automatic WIR/WIE system". The automatic WIR/WIE system is short for automatic Web Information Retrieval/Web Information Extraction system. For clarity, this system is referred to as Xhemali's automatic WIR/WIE system in the rest of the thesis. The system has been developed using VB.NET 2008 and it was designed to be used by the advisors (the users) at Apricot Training Management (ATM) who are responsible for providing the training course information to their clients.

Xhemali's automatic WIR/WIE has two components; Web Information Retrieval (automatic WIR module) and Web Information Extraction (automatic WIE module). These components are described in the following sections. Figure 3.4 shows the high-level view of the system's architecture.

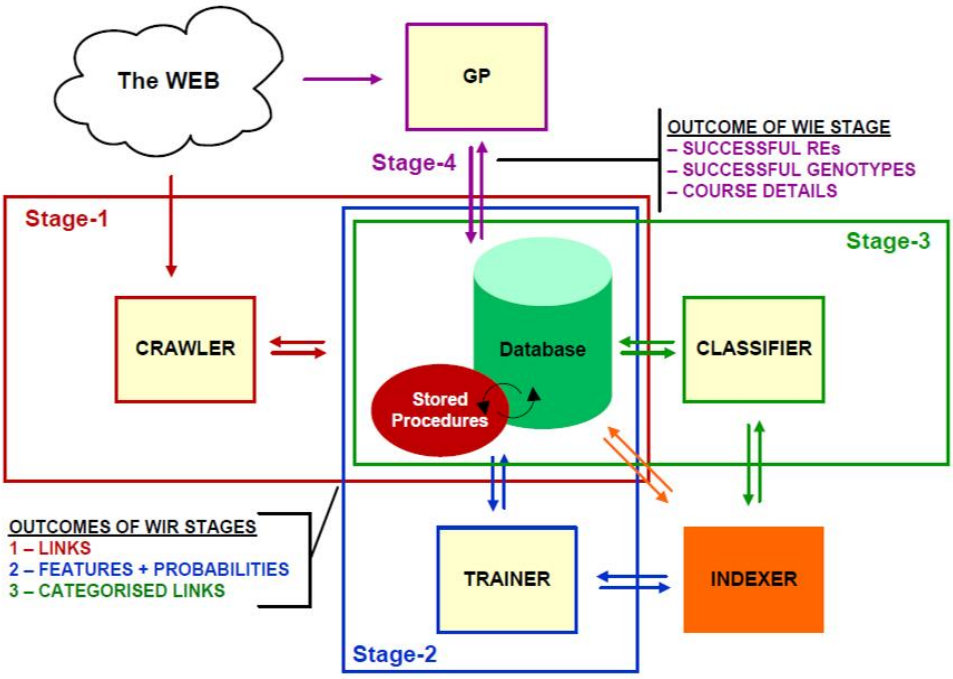


Figure 3.4 Xhemali's Automatic WIR/WIE System overview (source Xhemali 2010a).

3.3.2 Automatic WIR module

Xhemali's WIR module of the system serves as the mediator between the Web and the ATM's Customer Relationship Management (CRM). This module is responsible to search the Web for training courses websites by analysing and determining whether these websites are likely to be useful or not for ATM.

Xhemali's WIR module consists of four stages; Crawler, Trainer, Indexer, and Classifier. The first stage, which is the Crawler, will find and retrieve all training courses web pages, and store the URLs of these web pages in the database. The user needs to provide initial URLs to seed the crawler.

Next, the second stage, which is the Trainer, is responsible for analysing whether these web pages contain relevant information or not. The indexer's function is to extract appropriate and most frequent text tokens, which are initially stripped off from HTML codes including scripts, from each of the relevant web pages. The Trainer and Classifier use these tokens for future analysis. The Classifier function is to analyse and classify the previous web pages into relevant and irrelevant web page categories using the classification algorithms (Naïve Bayes approach). Based on the data collected from 24 websites consisting of 163,340 web pages, Xhemali claims that her method is better than Web Link Validator and Link Checker Pro. This crawler also outperformed Google in finding top ten leadership and management websites relevant to ATM.

This crawler is not within the focus of this thesis, other than using the URLs it collects from the UK training course domain that are relevant to this study.

3.3.3 Automatic WIE module

The web extractor (Xhemali's automatic WIE), which is the other part of the system, is an automatic wrapper. It is responsible for extracting four attributes of interest from the online training course information i.e. the course title,

location, price and date, from the course providers' domains in the UK. These web pages are crawled and retrieved earlier by Xhemali's WIR module.

Genetic Programming (GP) is used by this system to automatically evolve the regular expression. Regular expressions are generated using rules stored in the XML file for extracting the relevant course information. There is no user involvement at this point and the user does not need to be familiar with GP nor regular expression syntax. In the following section, Xhemali's automatic WIE components are described, followed by the processes of extraction in more detail.

Because web pages are human-oriented documents, which have various and irregular formats, the automatic extractor may not be able to recognise the information required. Xhemali reported that the results from her experiments on 60,000 'never seen before' web pages, shows that there is a need for the automatic extractor to automatically increment its extraction rules. Based on those experiments, the automatic web extractor has achieved an accuracy of over 94% for the extraction of course titles and an accuracy of just below 67% for the extraction of other course attributes which are dates, prices and locations. This provides a motivation to design a better solution, which provides additional relevant features (value added features) built onto the existing system to enhance its functionality such as automatic increment of the extraction rules. The addition of the human involvement to identify specific pieces of course information is presumed to make the web extractor become a more effective and comprehensive system.

Evolving Regular Expression

In Xhemali's automatic WIE, the extractor uses a set of extraction rules evolved by the GP system to define a set of extraction patterns (made up of regular expressions), then applies the pattern to the web page that is retrieved to find and capture each course attribute in focus. A set of domain-dependent extraction rules is available in the XML file. In the GP system, the Genotype-Phenotype mapping is used to build a regular expression with the aim of

finding the optimum extraction pattern, which when applied to each web page should extract the candidate extraction attributes. The regular expression creation is fully automatic in contrast to other pattern creation methods.

The example of extraction pattern captured for price attributes consists of three components; the tag information (pattern 1), the keyword and the data format (pattern 2). The information must match these components to qualify for the extraction. However, the fitness evaluation is only performed on Pattern 2, after Pattern 1 is first removed.

Pattern 1 (tags format) : <tr[\s]?id=”row1”.*?>[\s]?<td.*?>.*?</td>

Pattern 2 (data format): (price|cost|fee).*?(£|£)\b\d.*?\b(\.\d{2})?

The XML-based rules are described in the following section. Similar to the software evolution, the mapping uses the modulo operator to determine the structure of the regular expression to be formed.

XML-based representation of rules

XML (eXtensible Mark-up Language) is one of the prominent technologies to present data in a structured manner that can be manipulated by different types of applications. The advantage of using XML is that it provides a standard (Bray et al. 2008) for structured-document markup and is compatible with the majority of the programming languages.

XML-based representation of rules has been a growing interest and has been applied in the genotype to phenotype mapping (Barrero et al. 2010). This is due to the growing number of XML documents being used to store data in a defined manner that originated from different types of sources including the Web. The XML can represent rules in a hierarchical structure where each path provides a single rule, which can be made up of several related components.

Rules, which are used to aid the formation of a valid regular expression pattern in Xhemali's automatic WIE, are stored in an XML file (refer to Figure 3.5). Because they are built manually and her automatic WIE has no module to accommodate for the addition of new rules, a regular expression expert is needed to do the update. Rules in this file are separated into 4 categories:

i) the tags.

   The tags collection consists of an itemised list of HTML tags, such as 'td', 'p' and 'div'.

ii) the regular expression rules.

   The regular expression rules category defines some guidelines on how to produce a valid regular expression. For example, the first component is an open-tag (*'<'* symbol), followed by a tag name (one of the elements in the 'tags' category) and a close_tag (*'>'* symbol), then a start_capture (*'('* symbol) followed by regular expression substructure (one of the elements in regular expression substructure category) and the stop_capture (*')'* symbol) and an end-tag (*</ tag name >*). Thus the resultant regular expression would be *<td>(.*?)</td>*.

iii) the regular expression substructures.

   The regular expression substructure is a structure where some quantifiers ('*', '+', '?') are added after a token such as ".*?" to match zero or many characters and "[\s]?" to match any space if any.

iv) the keywords.

   The keywords component contains a list of keywords identifying the attributes, i.e., price, title, date and location.

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <rules>
    <rule id="0">
      <component id="0">9</component>
      <component id="1" no_end="true">4</component>
      <component id="2">3</component>
      <component id="3">10</component>
      <component id="4">3</component>
      <component id="5">4</component>
      <component id="6">6</component>
      <component id="7">3</component>
      <component id="8">7</component>
      <component id="9">5</component>
    </rule>
    <rule id="1">
      <component id="0">4</component>
      <component id="1">6</component>
      <component id="2">3</component>
      <component id="3">7</component>
      <component id="4">5</component>
    </rule>
  </rules>
  <restructures>
    <restructure id="0"><![CDATA[.*?]]></restructure>
    <restructure id="1"><![CDATA[[\s]?]]></restructure>
    <restructure id="2"><![CDATA[id="row1"]]></restructure>
  </restructures>
  <tags>
    <tag id="0"><![CDATA[title]]></tag>
    <tag id="1"><![CDATA[td]]></tag>
    <tag id="2"><![CDATA[tr]]></tag>
  </tags>
  <keywords>
    <keyword id="0"><![CDATA[name]]></keyword>
    <keyword id="1"><![CDATA[course]]></keyword>
    <keyword id="2"><![CDATA[title]]></keyword>
  </keywords>
</root>
```

Figure 3.5 XML-based grammar rules (source Xhemali et al. 2010b)

The following section describes in more detail how Xhemali's automatic WIE builds the extraction pattern for an attribute.

WIE processes

Xhemali's automatic WIE module aims to do the extraction process without any human involvement or requirement for a human expert to build the regular expressions and transfer the result as well as the extracted keywords to the database for future queries. Figure 3.6 shows the database structure developed for the system, which is not only used to store relevant training information extracted but also intermediate data to support the system's functions such as links, word count, link status, genotype and phenotype details. But before the extractor understands and captures the information, it is important for it to "know" how to recognise the information.

Xhemali's automatic WIE system relies heavily on a true match of the patterns of the data with the regular expression that it generates before the extraction task can be successful. It is built on GP principles to evolve the regular expression based on the set of extraction rules. The regular expression notations, which are stored as rules in an XML file, are manually created from the careful analysis of the relevant web pages. The rules are chosen and combined together based on Genotype-Phenotype transformation to form a valid regular expression. The extractor matches the resulting regular expression (evolved) to text in web pages and applies the fitness test.

A regular expression is successful when its fitness scores satisfy the set maximum value. This regular expression is stored away and can be reused to assist another extraction process. Regeneration of regular expressions normally happens whenever the stored regular expressions in the database fail to match data from the retrieved web page. This failure might be caused by structural changes made to the system's previously processed web pages or the system is processing a new web page. In such a case, the extractor will need to relearn those changes or the newly discovered patterns. However, this is not possible as the rules available to the system are fixed.

The following steps describe the processes of extraction:

1. **Analyse web pages**: Xhemali's automatic WIE analyses the previously retrieved web pages (of the relevant links) from the database (CIE_Allowed_links table) and applies the regular expression pattern to find the relevant information. If it fails to discover the new pattern, then regular expression evolution will begin (step 2).

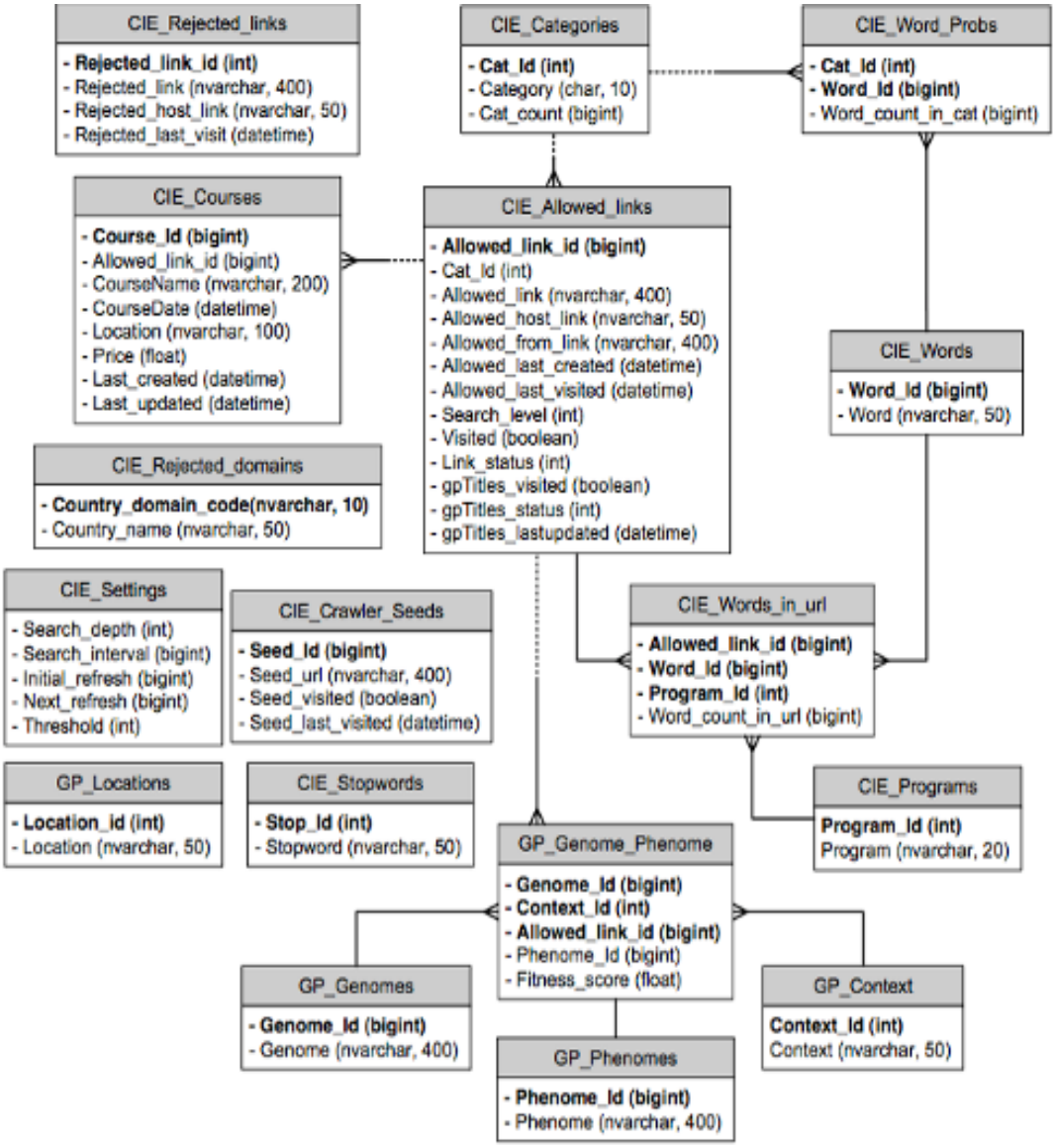


Figure 3.6 The database structure used by the automatic WIR/WIE system (source Xhemali 2010a).

2. **Regular expression evolution using GP**: This process is divided into five main tasks:

    a. *Generation of initial population.*

    An initial population consisting of ten genomes is randomly generated. Each genome is made up of variable length of genes and is translated to a phenome using genotype-phenotype mapping process. Each phenome is then executed and its output quality is measured using a fitness test function. Fitness, in this case, measures how well a generated solution has learned to predict the output from the input during the evolution. The fittest genomes will then be carried forward to the next generation for reproduction while the less fit ones will be discarded from the population.

    b. *Parent Selection.*

    Individuals in the current population are put through a tournament selection to choose the best individual for reproduction. For the selection of the first parent, 40% of the population are randomly chosen and their fitness scores are compared. Two individuals with the highest fitness scores are selected and will be used to create the next generation of individuals (offspring). This selection process is repeated for the next parent until the total of parent matches the size of the population, i.e., five parents.

    c. *Reproduction*

    During the reproduction process, the offspring are generated from the parent genomes with the aid of two genetic operators: uniform crossover and mutation. These offspring will become the new individual where each individual is translated to a regular expression using Genotype-Phenotype mapping process. Repairing function is then applied to the generated regular expression to ensure they are syntactically correct.

d. A fitness test

The fitness measurement is used to determine the validity of the regular expression to extract the course information from web pages. There are two fitness tests introduced; fitness test for the course title because it is always independent from the remaining attributes (course date, location and fee) and fitness test for these remaining course attributes because they are dependent on one another, hence, needed to be managed as a group.

Xhemali uses several criteria to evaluate the regular expression, which includes the formatting, geographical location, length of the extracted attributes, single versus set of attributes, page position, regular expression length and regular expression duplication. Each criterion contributes to the overall fitness score of individuals. Distinctively, a Naïve Bayes approach is applied to predict the usefulness of the evolved regular expressions for capturing the course title (Detail information can be found in a paper by Xhemali et al. (2009)). According to Xhemali (2010a), however, this Naïve Bayes method is not suitable for the rest of the attributes in focus due to partial dependency issue, where sometimes one attribute is dependent on the other, such as the price of the same course is determined by the date and the location.

If any offspring are found to be fitter than individuals in the parent population, it replaces the weakest individual for reproduction selection process, otherwise this offspring is rejected.

e. Termination of evolution

This GP process is terminated when it encounters any of the three conditions below:

i. it finds the perfect solution

ii. it has reached the specified maximum generation cycle

iii. no improvement in the solution for a specified number of generations, 100 in the case of Xhemali's work.

3. **Capture information**: The data, which matches the evolved regular expression pattern, will then be captured from the web page that has been retrieved and transferred data to the database. The extractor may capture a single instance or multiple instances from a single web page.

3.3.4 Discussion

From the analysis performed on Xhemali's automatic WIE system, it becomes apparent that this WIE system is less effective in the extraction of the three attributes out of the four (price, date and location). This provides the motivation to build a system that would be able to detect these attributes, through their unique features and location on the web page.

One of the main contributions of Xhemali's automatic WIE is the use of XML-based extraction rules. However, the XML-based extraction rules, while providing portability to adapt to new domain, are not very flexible in terms of recognising and extracting the relevant information from the unseen web pages. While manual addition of new rules could be an option, it is unlikely to provide an adequate solution as this requires some XML expertise and thorough understanding of the kind of rules to be added.

In summary, Xhemali's WIE system achieved the following:

1. The system has the capability of evolving the data patterns, which is in a form of regular expression. This means it can produce the unique patterns automatically that will match and extract the specific pieces of training course information. These generated patterns can be applied successfully to other similarly structured web pages; otherwise new patterns will be generated instead.

2. The extraction rules are stored in a separate XML file, which can be called by the GP program. As it is external to the GP program, the file

can be replaced to work on a different domain with minor modifications without disrupting the operation of the rest of this GP program.

3. The novel use of the XML technology to store the extraction rules provides a lot of benefits in terms of human readability, compatibility with other programming languages, portable to many operating systems and extensible to a deeper or wider hierarchy.

4. The system operates automatically with minimal user involvement. The user involvement is only needed at the beginning to start the execution.

5. The system can be used by other organisations with similar requirements to ATM. However, a few changes need to be done for different requirements or domains.

3.3.5 Critique

The following describes four significant weaknesses of Xhemali's automatic WIE system:

1. One of the most significant weaknesses of Xhemali's automatic WIE system is that sometimes it is unable to extract the course information because information moves location on the page or its format changes or it is no longer available. This can be improved further through semi-automation.

2. Although there was a thorough analysis of the structures of web pages and their contents to formulate the extraction rules, web pages are regularly updated and new technologies are introduced, which may introduce a new representation or new structural representation. Because these extraction rules are crafted manually by an expert it is unlikely to be successful on every web page. In addition, the system was not designed to accept new rules automatically and it highly depends on this expert to update it.

3. In order to produce a valid phenome by the Genotype-Phenotype mapping process, the Repairing function plays an important role in the current Xhemali's automatic WIE system. It is designed to work on the training courses websites and it would be very difficult to customise this function for new domains as the main controls of the rules representations are coded in the program, which requires a programmer.

4. Applying 40% selection technique for the tournament is a high proportion. This means that the probability of getting a very fit individual is high and the weaker individuals stand less of a chance of being selected.

3.4 Chapter Summary

This chapter present the study of THREE early works; TWO works on software evolutions and a work on automatic Web Information Retrieval/Web Information Extraction (WIR/WIE) system. It discussed the analysis undertaken to meet the third objective specified in Chapter 1; how these works can be used as a basis for developing a suitable WIE software tool for an improved WIE solution?

The study focuses on two works on software evolution; Withall (2003) and Xhemali (2010b). Four issues were found in these works, which includes difficulty of expanding rigid rules, restricted access to controlling functions, imperfect fitness measure and dependency on a repairing function. They are elaborated in Section 3.2.4 to show the motivation for an improvement. An important note is also included to justify the necessity to take up this area for this thesis.

The second section discusses the components of Xhemali's automatic WIR/WIE system and how they are inter-connected to produce a solution for Apricot Training Management (ATM). Several significant weaknesses of this system are also highlighted in Section 3.3.3, justifying the needs for the initiative to address them in this thesis. One of the most significant drawbacks

is that Xhemali's WIE system struggles to capture information from the previously processed web pages if these web pages experienced structure or content change. The extraction rules are hand coded and have limitations in immediate adaptation to new data thus hindering the extension of the extraction coverage. Another shortcoming is the embedded 'repairing function', which makes the system less flexible for future expansion and finally the high proportion selection used in the parent selection techniques is giving slim chances to the weaker individuals to be selected, instead of giving a fair chance to all individuals.

Details regarding the research development as well as the evaluation of the improved solution on evolving complete software and evolving regular expressions for the WIE system are presented in Chapter 4 and Chapter 5 respectively.

Chapter 4

Practical Application of GP – Domain 1

4.1 Chapter Overview

Chapters 2 and 3 extensively reviewed the literature and works related to the field of information extraction to provide a foundation for building a framework for the semi-automatic system with evolved extraction patterns. This chapter presents the approach and methods of evolving rules aiming to overcome the issues identified in Chapter 3 and to fulfil objective #4 of this thesis. Specifically, the intention is to demonstrate the enhancement to the Genetic Programming (GP) method to produce a design in support of high system efficacy (Section 4.3.3). When the generated regular expression is referred to be 'fully fit', this means the maximum fitness score for that individual is reached and satisfied all the criteria for the test source. However, this individual is not necessarily completely correct for all possible web pages.

Section 4.2 introduces the context of the research; computer program evolution and regular expression evolution. The next section details the design and development of the evolutionary systems. Section 4.3.3 introduces the enhanced GP mapping method and the grammar to support this mapping. The efficiency of the evolved program is measured by the fitness function, which ensures that the correct program solution is achieved. The evolved program needs to be 'fully fit' to be useful.

Choosing a suitable programming language is important for developing an evolutionary system and the reasons why PHP (the original choice) was eventually not suitable for this project are also provided in Section 4.3.4. In Section 4.3.5, the general experiment set up which is applicable to both evolutionary systems is presented.

4.2 Context

This section describes the context in which the Evolutionary System was developed; a complete program evolution (this chapter) and a regular expression evolution (Chapter 5). The aim is to produce a new method that improves the performance of the system in both domains.

The first few sets of experiments aimed to find an optimal solution for complete program problems. Optimisation is loosely defined as a process that finds a better or a total optimal solution to a problem in a particular domain according to the optimality criterion (Poole & Mackworth, 2010). It was demonstrated that the technique presented here was as efficient as the earlier techniques by Withall (2003) and Xhemali (2010a, 2010b), and outperformed in certain parts. The next experiments concern the production of regular expressions, which are successfully evolved to extract attributes, i.e., course title, start date, location and price from the training course websites.

Apart from developing the GP system, a special script is also included. This script allows for the automatic running of experiments with a fixed set of random number seeds and automatically stores the results of the experiments. This means execution of the new GPs can start immediately after the previous one is completed, thus the stored results can be analysed at a different time.

Like the regular expression evolution, the fundamental target schema of the WIE system developed in this research (Chapter 5), is to extract from training course web pages with explicit but not necessarily all the available attributes. Figure 4.1 depicts the route of the experiments of the tasks involved in this research and how they are related or connected.

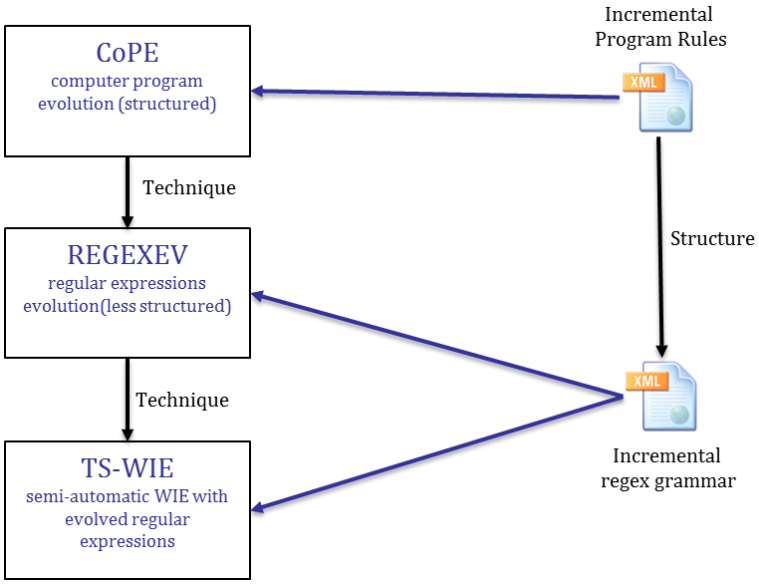


Figure 4.1. Experimental Route showing the progression of the tasks in this research and the relationship between the components.

4.3 Evolutionary System Approach

4.3.1 Introduction

Two distinct evolutionary systems are described here; software and regular expressions evolutionary systems, which are inspired by the work of Withall (2003) and Xhemali (2010a, 2010b) respectively (see chapter 3). Both earlier systems used Genetic Programming (GP) as the evolution strategy to solve the problems. One of the key components of interest in their strategy is the Genotype to Phenotype Mapping method. Separation of genotype and phenotype has provided flexibility to apply genetic operators such as mutation and crossover. This method of separation is the work of Banzhaf (1994), who demonstrated efficient results. Some other researchers such as Paterson and Livesey (1996), and O'Neill and Ryan (2003) also proved the same. However, this separation also brings in some difficulties during the mapping process such as shortage of genes, inheritance of characteristics and maintaining a syntactically correct program, thus it is important to establish a direct and consistent mapping (Withall et al. 2009).

This section addresses these difficulties and introduces an efficient way to avoid them. To prove the technique that enhanced the GP mapping method, the experiments in this research began with an evolution of a software program. The syntax rules of a programming language are rigid and structured, thus the data used for the input can be controlled easily and the output is predictable. A similar approach was then used to evolve regular expressions, where the syntax is much more complex and less structured.

Before describing the separate experiments in detail some features of the evolutionary system common to all experiments are given.

4.3.2 Grammar Validation Tool

Apart from its compatibility with major programming languages and platform portability, XML is preferred in this research as the XML tags are not predefined, thus the developer has more control over the process of describing the rules of various levels of complexity. Successful use of XML to support the transformation of phenotype from genotype has been recently demonstrated in the work of researchers like Xhemali et al. (2010b) and Barerro et al. (2010). The rules are written in an external file, thus it is globally available and easily replaceable to work for different requirements or other domain with only minor interruption to the main evolutionary system. This flexibility is demonstrated in Section 4.4. and Section 4.5.

The rules used in the program evolution are designed as a “well formed” and “valid” XML document, validated against a Document Type Definition (DTD). A DTD dictates what elements should appear where in the XML document, including what elements and attributes may be contained in each of these declared elements. The DTD may also be used by an external system to parse the XML document. In this thesis, DTD is applied to ensure the correctness of the XML structure and it is an integral part placed at the top of the XML document.

A special DTD was created for this research to ensure that the XML document conforms to the DTD rules and it is illustrated in Figure 4.2. It can be interpreted as the root element is *grammar* and it has two elements; *start* and *rules*. The start consists of a single non-terminal element and the rules contain zero or many elements labelled as *rule*. Note that the '*' symbol indicates that an element happens zero or many times. Each rule contains either a non-terminal or token, which occurs either zero or many times. A non-terminal element has EMPTY, which specifies that this element must not have any elements at all, i.e., text elements or children elements. The token is type "#PCDATA", which basically says that it contains text data. The following ATTLIST means that a particular rule has attributes. In this case, each rule has two compulsory attributes as indicated by #REQUIRED; a type, which states that a particular attribute is a selection or a sequence, followed by a name to identify the rule. Finally, the only attribute of a non-terminal is a mandatory name.

```
<!DOCTYPE grammar [
  <!ELEMENT grammar (start,rules)>
  <!ELEMENT start (nonterminal)>
  <!ELEMENT rules (rule*)>
  <!ELEMENT rule (nonterminal|token)*>
  <!ELEMENT nonterminal EMPTY>
  <!ELEMENT token (#PCDATA)>
  <!ATTLIST rule
      type (selection|sequence) #REQUIRED
      name NMTOKEN #REQUIRED
  >
  <!ATTLIST nonterminal
      name NMTOKEN #REQUIRED
  >
] >
```

Figure 4.2 A DTD that defines the structure of XML-based rules with legal building blocks (elements and attributes). The DTD is declared within the XML document as an internal subset and it ensures compatibility with the XML systems.

The XML-based rules are derived based on the sequence of items either terminals (tokens) or non-terminals according to the programming syntax and following the DTD. For example, if the syntax of an IF statement described in BNF is

ifstatement ::= "if" "(" exp ")" statement

then this would be coded in XML as

```
<rule name="ifstatement" type="sequence">
    <token>if</token>
    <token>(</token>
    <nonterminal name="exp" />
     <token>)</token>
    <nonterminal name="statement" />
</rule>
```

4.3.3 Genotype to Phenotype Mapping Method

In this research, an evolutionary system is developed with the application of GP and XML-based grammar definition. One distinctive approach introduced is to find the number of non-terminals in a particular rule that defines the block size of a genotype and providing ordinal number reference to the non-terminals for the mapping process. The first gene of a block is always reserved for the decision of the type of rules to follow. Therefore the size of a block is first gene plus maximum number non-terminals (1 + NT). The system is then evaluated using an experimental method and the performance measured using the time it takes to reach a fully fit (successful) solution. The genotype (encoded program) to phenotype (program statement) mapping used in this research is described as follows:

Finding the maximum number of non-terminals in a production:

The mapping is heavily dependent on a particular number, which is the maximum number of non-terminals to be found in any production in the grammar. The formal notation used to make this notion precise is described ahead of the main mapping.

A genome is decoded into a phenome using a full syntax grammar definition. A grammar G is represented as a 4-tuple:

$$G = (N,T,NP,S)$$

where

N is the set of Non-Terminals

T is the set of Terminals

NP is a numbered set of productions - a set of pairs
(PosInt × Production)
S is the start symbol, $S \in N$

Unlike normal notation where NP is just a set of Productions, here it refers to the productions from a particular non-terminal by ordinal number. The set of all productions defining a particular non-terminal (n) can be discovered using $Productions(n)=\{ (?,p) : (?,p) \in NP \wedge p = (n ::= ??)\}$. Here, $?$ is an integer representing the ordinal number given to the production and $n ::= ??$ is a production where $??$ is any mixture of non-terminals and terminals.

All the productions from the same non-terminal are to be numbered sequentially from 0, so $/Productions(n)/ = m \wedge (i,?) \in Productions(n) \Rightarrow 0 <= i < m$. If $(i,p) \in P$ then p has the form $L ::= R$ and $L \in N$ and $R \in (N \cup T)^*$ [alternatively, using powerset notation $R \in \wp(N \cup T)$].

Consider one production p written as $L ::= R$, where R is a string of values of length r, written as $r= /R/$. Each element of R can be referenced by indexing, $Rj \text{ for } j=0 \text{ to } r-1$. The set of indexes is then defined for which R_j is a non-terminal $NTIdx(R) = \{j:R_j \in N\}$ and the number of non-terminals in R is counted $NTCount(R) = /NTIdx(R)/$.The set of all counts of non-terminals mentioned on the right hand side of a production P is $NTCounts(P) = \{c : c = NTCount(R) \wedge (?, ? ::= R) \in P\}$. Thus the maximum number of non-terminals among all the productions of a grammar can be found using $max(NTCounts(P))$, where $[m=max(S) \Rightarrow m \in S \text{ and } \forall e \in S, e <= m]$.

Take an *if statement* rule example from the previous section as an example. Figure 4.3 shows the set of indices of elements in this rule (R_j). The rule R, in this case, refers to the 'ifstatement'. Therefore,

NTIdx(ifstatement) = {2, 4}, and
NTCount(R) = |{2,4}| is 2

which indicates that the size of the 'ifstatement' block is 2. The purpose is to find the maximum size to determine the size of the blocks in the genome.

IF statement – XML format	Index (R_j)
<rule name="ifstatement" type="sequence">	
<token>if</token>	0
<token>(</token>	1
<nonterminal name="exp" />	2
<token>)</token>	3
<nonterminal name="statement" />	4
</rule>	

Figure 4.3. Index of elements in 'if statement' rule of type 'sequence'.

The above example showed the process of finding the size of a rule of type "sequence" and this NTCount(R) has a different purpose to a "selection" rule.

Take a rule called "statement" which contains several options of statements as represented in BNF as

statement ::= ifstatement | for | assign | add | subtract

and its equivalent in XML form is

```
<rule name="statement" type="selection">
    <nonterminal name="ifstatement" />
    <nonterminal name="for" />
    <nonterminal name="assign" />
    <nonterminal name="add" />
    <nonterminal name="subtract" />
</rule>
```

To calculate the number of genes (size of the block) required for this rule (Figure 4.4 shows how the index is assigned to each non-terminal):

NTIdx(statement) = {0, 1, 2, 3, 4}, and

NTCount(R) = $|\{0, 1, 2, 3, 4\}|$ is 5, therefore

the size of 'statement' rule is 5 and it is useful to find the numbers of available choices for this production rule to decide which option to take and this is defined by taking the modulo of a particular gene value. For example, if the gene value was 10 then 10 mod 5 = 0, so the statement with index 0 (first statement) would be chosen.

statement – XML format	*Index (R_j)*
<rule name="statement" type="selection">	
<nonterminal name="ifstatement" />	*0*
<nonterminal name="for" />	*1*
<nonterminal name="assign" />	*2*
<nonterminal name="add" />	*3*
<nonterminal name="subtract" />	*4*
</rule>	

Figure 4.4. Index of elements in 'statement' rule of type 'selection'.

Applying the result:

Now the maximum number of non-terminals concept is used in defining the genotype to phenotype mapping. It defines the number of genes per block (b) in the genotype as $b = max(NTCounts(P))$. Because the genes appear as integer codes, if a non-terminal with name n is expected and an integer code i is given, then production to be used is $p = (k, n ::= ?)$ *where* $k = i$ *modulo* $/Productions(n)/$.

The genotype-phenotype translation algorithm is expressed as follows:

A genotype (GT) is a sequence of blocks ($B_0B_1...B_{n-1}$) for some n. Each block (B) is a sequence of genes ($g_0g_1...g_{b-1}$). Each block records the encoding of one production (p). If $p = (i, L::=R)$ then integer codes are given for each R_j in turn for which $R_j \in N$. Note that no codes are given where $R_j \in T$ because there is no choice as the terminal must be included. The integer code for the non-terminal case chooses which of the relevant productions is to be expanded.

If the encoding process yields less than $b = max(NTCounts(P))$ integers, then extra arbitrary genes are added by padding on the right in order to keep all blocks the same length (Withall 2003). Genes added in this way are never used in the decoding process but they serve a crucial purpose in the generation/mutation process.

4.3.4 Programming Language

Initially, PHP 4 was chosen as the programming language to build the evolutionary system. The main reason was that the much of the other parts of the system were written in this language. Other reasons are that PHP is free and open source software, can be used with any Relational Database Management System (RDBMS) and is suitable for web applications. In 2011, PHP was ranked #4 in the TIOBE Community Programming Index (TIOBE, 2011). New releases and patches are issued on a regular basis thus fixes can be done much faster. Another reason for choosing PHP was that it is a scripting language with a built-in library that has a wide variety of functions, and it can handle pattern matching well (this is particularly useful in the extraction process) using its built in commands such as *preg_match(), preg_replace()*. If the evolution program fails to work properly, such as trying to perform an infinite loop, PHP can handle this kind of error by giving a specific time to each program before it is terminated and this process does not affect the rest of the GP process. Moreover, the resulting program, which is internal to the GP process is also written in PHP, thus no other compiler is required to be installed in this situation.

In practice, the computing resources requirement, especially memory consumption, is one of the disadvantages of GP (Walker 2001). From the researcher's experiments of software evolution, which involved tens of thousands of loops, it was found that the weakness of PHP 4 became apparent when managing memory. This consumption could grow very quickly and substantially in order to evolve the solution causing the speed to gradually slow down and eventually the program crashed. Several attempts to find out which part of the program consumes the most memory and then optimise this memory usage, such as refreshing memory every one thousand cycles, reassign a null value to a variable instead of unset and use functions where appropriate, were ultimately unsuccessful.

In the latest version of PHP (PHP 5), the PHP developers claim that they have addressed this problem of memory management by introducing the garbage

collector to clean-up the unused variables when the root buffer for holding these variables is full (PHP.NET 2010). This collector should prevent the creation of memory leaks as the run progresses, so the experiments were re-run in PHP 5. However, the slow performance issue was still unresolved. This is because the garbage collector is only called in when the program requires more memory, thus the release is not quick enough for the execution to progress. The only advantage observed was that it avoids the program crashing as a result of insufficient memory allocated. Figure 4.5 shows the empirical evidence of the performance of the evolutionary system developed in PHP from the start of execution to 7000 generations/cycles. So, a new programming language was sought.

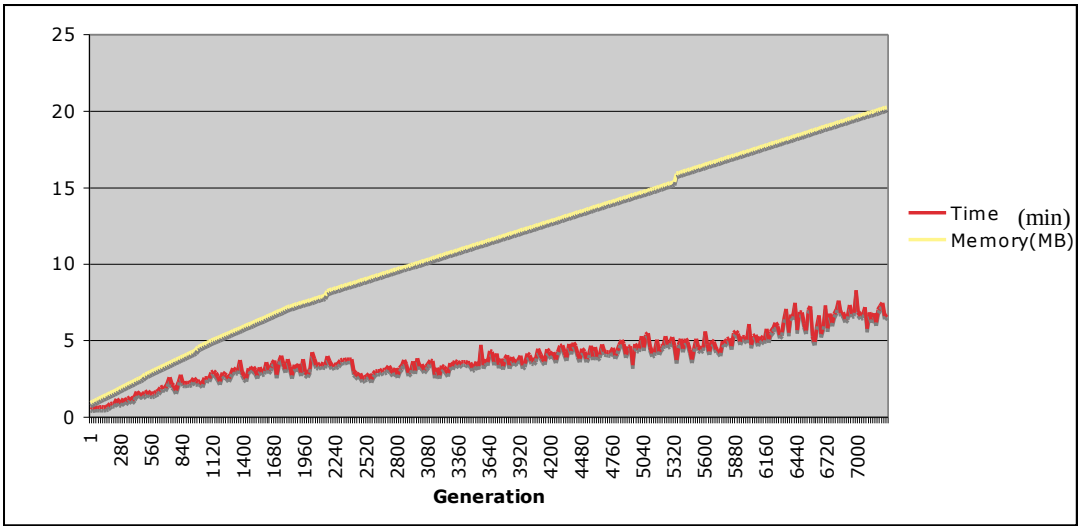


Figure 4.5 Time vs memory consumptions by the evolutionary system in PHP.

The scripting language PERL is one of the most powerful tools in text processing and excellent at handling regular expressions (Pham & Wilamowskl 2009). Like PHP, PERL is an interpreted language, which has the benefit of smaller executable program size and is normally platform independent. Although the codes are parsed and executed at runtime, which makes it slower in terms of processing speed, interpreted language is better for artificially evolving programs as the execution time is faster. The following are the main reasons for choosing PERL:

1. It has some aspects of functional programming, which the developer is most comfortable with, thus debugging is easier.

2. It has a huge collection of libraries, which are frequently maintained and they are downloadable from CPAN.
3. Availability of good features for Artificial Intelligence programming such as garbage collection, extensibility and interactive environment.
4. It has equivalent or even superior pattern matching than PHP.
5. It has debugging tools and programming support, which allows the developer to draw resources from the community.
6. It has a good interaction feature with the Internet using modules such as *LWP::Simple* and *Web::Query* to handle HTML web pages, which is made simple by the CGI.pm module, thus useful for this research.

Because of these reasons, the GP program was re-coded in PERL. For execution up to 7,000 cycles, the computational effort was halved, knocking down the execution time from more than 8 minutes to just above 4 minutes and a great improvement on the overall memory usage from 19MB to just 1.4MB.

As mentioned above, this thesis is exploring evolutionary programming subsets in solving computer problems and evolution of regular expression in Information Extraction. The following outlines the experimental set up including some general parameters applied to both areas; computer program and regular expression evolution.

4.3.5 Test Environment

All experiments are carried out using the following general parameters and Table 4.1 shows the settings for each of the experiment tasks:

Execution Condition: The system executes ten runs; each evolves the population for a maximum of 50,000 generations with ten different random seeds based on the first ten prime numbers (1, 2, 3, 5, 7, 11, 13, 17, 19 and 23). A random seed means the identical sequences of

integers are generated every time so that the run can be made repeatable. Withall and Xhemali (see Chapter 3) only need this maximum setting to produce good results. Therefore, to make a fair comparison, the same parameter settings were used.

Population settings: In the initial experiments, small tests using various initial population sizes were tested to determine the best setting for a specific problem to solve. Then the most promising population size was selected based on the size, which has the smallest average generation and the lowest average speed requirement. In the first few experiments, the most common method is applied where the individuals were randomly generated to fill the initial population and in the remaining experiments, populations were seeded with known solutions.

Representation: Each genome is made up of a string of integers called genes. The gene is not restricted to the integer data type. It can be of any data type such as real numbers and binary strings, but to be consistent with the previous works, integers were chosen here. The genome length was using fixed-block lengths and the size of block was determined by the rule, which requires the largest number of non-terminals.

Parents Selection: The selection method was using Roulette Wheel Selection. From experiments, the best result was achieved using this approach.

Genetic Operators: Uniform crossover is applied with a probability set to 50%. The mutation rate was set to 10%. This was the best rate from experiment and this low mutation rate aimed to make small random changes to explore new possibilities in the search space and to sustain the convergence of GP.

Fitness Test: In order to consider whether an individual is fit or not, the fitness score must be 100% of the predefined target fit value. The fitness score is calculated based on the weight set for the evaluation elements (sometimes called fitness cases) that specify the desired goal of the search for a fit solution process. The higher the value of the score achieved by a solution, the fitter the individual. A specific

fitness evaluation weight set is needed for other computer program problems or other domains.

Termination Condition: The experiment terminates if it has reached the specified maximum generations (50,000 generations) or it has found a fully fit solution.

Machine Specification: All experiments use Intel 3.00GHz PC with 4GB of RAM, running Windows7, therefore, the speed recorded is based on this specification to ensure consistency.

Table 4.1 The characteristics of the experiments carried out in this research.

	Common Characteristics	
	- Random number set for initial population - 50% crossover rate - 10% mutation rate - max 50,000 generation cycle	
#	**CoPE**	**Special Characteristics**
1	Sorting lists of integers	7 genomes
2	(Seeded) Sorting lists of integers	7 genomes seeded with previously successful evolved genome
3	Reverse-sort lists of integers	7 genomes with modular approach
4	Distance-from-mean	7 genomes
#	**REGEXEV**	**Special Characteristics**
5	REGEXEV with Extraction rules	10 genomes
#	**TS-WIE (Chapter 5)**	**Special Characteristics**
6	Regular expression with automatically incremented grammar using target web pages	10 genomes
7	(Seeded) Regular expression with automatically incremented grammar using target web pages	10 genomes seeded with previously successful evolved genome

4.4 A Computer Program Evolution (CoPE)

4.4.1 Introduction

Applying a fixed-length block genome technique helps to maintain the characteristics of the parents in the children, with very small effect on the following blocks if an earlier block experiences any change due to mutation. The size of blocks (*BS*) corresponds to the grammar rule, which requires the most information, i.e. the highest number of non-terminals (*NT*) plus an extra gene that is placed as the first integer in each block to decide the type of rule to follow, $BS = NT + 1$. Each block translates to a syntactically correct code in the phenome, therefore the completeness of the generated program can be maintained to allow for a proper execution of the program without interruption. In addition, a 'clean' grammar is introduced to support the mapping method, guaranteeing that a valid program is produced. A 'clean' grammar is defined here as a grammar that follows the correct syntax of a particular programming language to produce an error-free program. A detailed description can be found in the following sections.

Aim of Experiment

The CoPE (**Co**mputer **P**rogram **E**volution) experiment was conducted to establish an effective method for handling the problem of gene insufficiency, maintaining characteristics inheritance and maintaining a syntactically correct program without using a 'repair function'. A 'repair function' refers to a mechanism to tackle syntactical errors in the program and force it to be a valid and runnable program (Banzhaf 1994; Ryan et al. 1998).

Subject

The GP system is designed to evolve a complete and error-free program. The specific programs chosen to demonstrate the system are a sort program, a reverse-sort program and a distance from mean program. The system accepts some lists of integers of various lengths as an input. The sort program sorts each list into ascending order, while the reverse-sort program is the opposite, arranging the integers in each list in descending order. The

distance from mean program is to calculate how far each integer in the list is from the mean of the list. The sort program was used by both Withall (2003) and Xhemali (2010a) as part of their work so it is appropriate to use those targets for a reference performance. The distance from mean program would require two blocks of code, which individually need terminating brackets and so is likely to favour the clean grammar approach.

Specific Setting for all CoPE problems

i. The population size: 7 genomes.
ii. Genome size : 10 blocks with 5 genes in each block
iii. Input : Lists of Integers stored in an array = { {30,40,60}, {60,85,75}, {90,93,95,98}, {90,89,85,57}, {40,45,48,39}, {20,30}, {40,30,35,39}, {50,6}, {30,28,29}, {50} }.
iv. Programming Language : PERL v 5.14.2

4.4.2 Sorting Lists of Integers

This section presents a viable approach to automatically evolve a 'sorting program' and how the system can be optimised. It also reports a novel approach that improves the structure of the grammar, which guides the mapping process. One of the key distinctions is that this experiment uses a more comprehensive grammar rather than the simply defined language subset used by Withall (2003) and Xhemali (2010b). The problem with their grammar is that it was capable of generating a syntactically incorrect program. Therefore to avoid this situation, they introduced a fixing tool to ensure that the generated program follows the correct syntax. Furthermore, because their system was not designed to accept new rules and the grammar is built focusing on the requirement of the specified problem, the variation of solutions that can be produced is limited and this also limits the kind of problem that may be able to be solved.

In contrast to Withall who wrote the grammar (programming) rules as part of the source code, the approach by Xhemali, i.e., using an XML file external to

the program is more favourable. The grammar in this XML format can be arranged in the hierarchy of rules and elements, thus grammar maintenance is easier, which this research is intended to achieve. However, Xhemali's XML grammar needs to be improved by redesigning the XML representation. The reasons for this improvement will be discussed further in the 'clean grammar' section below. Therefore, the focus of this section is to investigate how GP methods can contribute towards enhanced methods of WIE and the evaluation shows the improved GP performance against the GP methods proposed by these two researchers. This provides a fundamental foundation towards developing a solution for WIE of training course data.

A sorting program or its opposite reverse-sort program was chosen for the experiment because it is one of many challenging computer applications, and is normally practiced by students in a computer studies programme. 'Sorting' either in ascending or descending order requires the use of selection, iteration and sequence statements, which comprise the basic concepts of programming. There are various sorting techniques available, for example, bubble sort, heap sort, and insertion sort, but for this experiment, the aim is to generate any working sorting program using a predetermined and modular programming syntax and a fitness function that recognises a properly sorted list.

Fitness Function

Fitness is calculated to determine how close the actual output produced by a particular phenome is to the expected one. This also determines how good an individual in a population is at solving a particular problem. It is also used to aid the selection of parents for the reproduction stage. The criteria used in the fitness calculation should be carefully chosen to accurately measure the ability of the individual at solving the given problem. A fitter phenome normally has a higher score and in most cases, a perfect individual is found if it reaches the set score, thus terminating the evolution process.

The GP system here is designed in such a way that it can be used for other

computer problems. However, a fitness evaluation function must be supplied for each problem. For the sorting and reverse-sort experiments, a fitness function is calculated based on two criteria; input/output pairs and the order of integers to determine the quality of the output list. A penalty is imposed if the output is empty or the execution took longer than the maximum time allowed, in which case, a minimum fitness score is given. The total score is then calculated. Therefore, the higher the fitness score, the more likely this genome will be carried forward to the next reproduction cycle.

A list is assumed to be sorted if its elements are in order, small to large. Figure 4.6 is a formal specification of a sort. In words, it describes that L is the starting list and N is the result. N is sorted if all elements in N are exactly the same as L, including duplicates, and N elements are in ascending order. It is written out as an all possible pairs test. A detailed explanation of this formal specification can be found in Cooke (2004). Because the formal specification of a sort is time consuming as the cost of calculation is O(i^2) where i the length of the list due to its tests on all possible pairs of integers, the fitness function applied here is calculated using a simpler version of the formal specification by Withall (2003). This simplified evaluation, which is O(i) is based on comparison of adjacent elements in the list, and every conjunctive goal will contribute to the fitness score. Unlike the traditional fitness function, which uses simple input/out pairs, this simpler version, shown in Figure 4.7, is shown to have better performance in Withall's (2003).

$$
\begin{array}{l}
sort : \mathbb{Z}^* \rightarrow \mathbb{Z}^* \\
\text{pre-}sort(L) \triangleq True \\
\text{post-}sort(L, N) \triangleq bag\_of(N) = bag\_of(L) \wedge ascending(N) \\
\quad \text{where } bag\_of(\langle\rangle) \blacktriangleleft\blacktriangleright \emptyset \\
\qquad\quad bag\_of(\langle x \rangle) \blacktriangleleft\blacktriangleright \{|x|\} \\
\qquad\quad bag\_of(L_1 \frown L_2) \blacktriangleleft\blacktriangleright bag\_of(L_1) \uplus bag\_of(L_2) \\
\quad ascending(N) \triangleq (\forall x, y : \mathbb{Z})(x \text{ before } y \text{ in } N \Rightarrow x \leq y) \\
\text{and} \\
\quad x \text{ before } y \text{ in } N \triangleq (\exists N_1, N_2, N_3 : \mathbb{Z}^*)(N = N_1 \frown \langle x \rangle \frown N_2 \frown \langle y \rangle \frown N_3)
\end{array}
$$

Figure 4.6. Formal specification of sort (source Cooke, J. 2004)

```
$fitness++   if(bageq(\@L, \@N));
if ($#N > 0) {
  for my $x (0..$#N-1) {
     $fitness++   if($N[$x] <= $N[($x+1)]);
  }
}
```

Figure 4.7. Simplified fitness function for sort (source Withall 2003)

Programming Languages like PERL and Java provide an enormous language library. As a good programming practice, only selective language subsets should be included in the program to solve a particular problem. The following subsection describes a novel technique introduced in this thesis, i.e., the 'clean grammar' which represent the relevant programming language subsets. But first, to help make the point clearer, the following discusses the grammar subsets.

The Grammar Subsets

It is commonplace in research on generating programs to work with a subset of a general purpose grammar. The reason is to restrict the search space of the genetic evolution to obtain answers more quickly. Researchers like Withall (2003) and Xhemali (2010a), also make the same point. However, the programming language subsets introduced in their work were not properly constructed. The main body of the conditional statements and their end statements, i.e. '}' in PERL syntax, are defined separately. This improper syntax could easily cause three syntax error situations; the end statement is the first line of code in the program, the conditional statements have it missing or there are too many. Therefore, to prevent from having this error, both researchers dealt with it via a repair function before the program is tested for fitness.

It is easy to subset a grammar where this just means deleting whole rules from the official full grammar, e.g., removing declaration and call of functions or removing one kind of iteration. It is more tricky to subset a grammar to restrict a number of constants and/or variables because new rules need to be written, e.g., var ::= "var1" | "var2" | "var3". Specifically for this research a new

variation on syntax has been invented. This is to request a repeat of an earlier identifier (further explanation is in the next section).

A 'Clean Grammar' approach

It is possible that the phenotype produced from the raw mapping of the genotype contains errors or incomplete elements to make up a valid program statement. This happens because individuals run out of genes required by a particular rule definition. To tackle this issue, researchers such as Banzhaf (1994) and Ryan et al. (1998) introduced a correcting mechanism, which is referred to as a 'repair function' throughout this thesis.

The same approach has been applied in both Withall's (2003) and Xhemali's (2010a) systems - the reason being that the generated programs are prone to have syntactically incorrect code segments. This repair function is executed after all the genes have been decoded. One of the purposes of this function is to insert the missing close brackets automatically to match the open brackets (Withall, or the 'endif' and 'endfor' in the case of Xhemali) in the generated program to allow for smooth program execution without interruption. However, this is strongly dependent on the specific evolution program, thus it needs to be duplicated for other programs.

This research proposes a 'clean grammar' to avoid such dependency on a 'repair function', which is specifically hand coded to handle specific problems while achieving an error-free program. A clean grammar is defined here as a concise representation of grammar which is hierarchically structured that

- follows the correct programming syntax construct,
- properly terminates a block structure,
- defines the rule's type : selection/sequence,
- defines distinction between each element of the rules; a non-terminal or a token (terminal),
- does not depend on a repair function to produce an error-free evolved program.

Figure 4.8 shows the 'clean grammar' represented in BNF form and its implementation in XML in Figure 4.9. The full grammar in XML is available in Appendix 3. The NULL statement is included in the grammar to provide an empty code that does not have any effect on the program produced as the number of code lines required by an optimum solution may vary.

statementseq	*::=*	*statement | statements*
statements	*::=*	*statement statementseq*
statement	*::=*	*nullstatement | assignstatement | ifstatement | forstatement | nestedforstatement*
nullstatement	*::=*	*";"*
assignstatement	*::=*	*wvar "=" rvar*
ifstatement	*::=*	*"If" "(" wvar opr wvar "{" statementseq "}"*
forstatement	*::=*	*"for" "(" cntr "=" 0 ";" "N1" "<" length ";" "N1" "++" ")" "{" statementseq "}"*
nestedforstatement	*::=*	*"for" "(" cntr "=" 0 ";" "N1" "<" length ";" "N1" "++" ")" "{" "for" "(" cntr "=" "N2" "+" 1 ";" "N2" "<" length ";" "N2" "++" ")" "{" statementseq "}" "}"*
wvar	*::=*	*"a[tmp1]" | "a[tmp2]" | "tmp3" | "tmp4"*
rvar	*::=*	*"a[tmp1]" | "a[tmp2]" | "tmp1" | "tmp2" | "tmp3" | "tmp4"*
op	*::=*	*"==" | "!=" | ">" | "<" | ">=" | "<="*
length	*::=*	*"length"*
cntr	*::=*	*"tmp1" | "tmp2"*

Figure 4.8. Grammar rules are expressed in BNF form. In the actual implementation, these grammar rules are presented in XML format.

For this CoPE research, basic grammar rules (primitive rules) following the correct programming syntax are manually coded in XML files. These rules represent the standard structure of loop statements (for and double for), an if statement and an assignment statement. Some of the rules are precisely constructed, such as, a for statement is represented as for (var1 = 0; var1 < length; var1++). The decision to implement a restricted construct is to reduce the search space and to ensure the validity of a statement constructed, thus speeding up the processing time without interruption. This means that some knowledge of the construction of a particular program to solve a particular problem provides an advantage to achieve an improved efficiency of the algorithm. For example, a 'double for' is a common structure used in the sort algorithms for comparing elements in a list. However, it was observed that not all solutions took advantage of the 'double for' statement.

```
<grammar>
 <start>
      <nonterminal name="statement" />
 </start>
 <rules>
   <rule name="statement" type="selection">
      <nonterminal name="nullstatement" />
      <nonterminal name="assignstatement" />
      <nonterminal name="ifstatement" />
      <nonterminal name="forstatement" />
      <nonterminal name="nestedforstatement" />
   </rule>
   <rule name="nullstatement" type="sequence">
      <token>;</token>
   </rule>
   <rule name="forstatement" type="sequence">
     <token>for</token>
     <nonterminal name="counter" />
     <token>(</token>
     <token>0</token>
     <token>..</token>
     <token>$#inlist</token>
     <token>)</token>
     <token>{</token>
     <token>$runtime++;  die if($runtime > $timeout);</token>
     <nonterminal name="statementseq" />
     <token>}</token>
   </rule>
  …….
 </rules>
</grammar>
```

Figure 4.9. An extract of XML-based grammar to guide the transformation of genotype to phenotype for generating a PERL program.

A few additional rules are also added to the grammar such as *statementseq* to define a block statement, to allow for syntactically correct statements generated and some rules in Withall's and Xhemali's solution are removed such as the *end statement* because they have no part in the clean grammar.

A special feature has been introduced in the grammar, which indicates a back reference to the previous non-terminals. They are labelled as 'N1' and 'N2'. This can be viewed at as a stack programming, operating a Last-in-First-out method. This feature simply tells the system that the non-terminal name for its replacement refers to the most recent non-terminal name being used. This approach is useful to simplify a loop statement construction, for example a valid for loop code, *for (var1 = 0; var1 < length; var1++)*, the non-terminal (in this case refers to a variable) '*var1'* appears 3 times; to initiate var1, to set the

condition for looping and increment the value of var1. In order to get the correct repeat of var1, it is replaced by N1. Therefore, this 'for' statement is represented as *for (var1 = 0; N1 < length; N1++).*

One of the benefits of this improved grammar definition is that a block of statements can be explicitly defined, automatically enclosed within the open and close brackets. The grammar not only allows for the creation of single statements, multi-block statements or nested statements in the program but also a mixture of these as a human programmer would do. Another benefit is that because the rules strictly follow the correct syntax, the generated program is guaranteed valid. This means no repairing is required. The third benefit is that the special DTD ensures that the valid structure of the XML file can be maintained. This will make sure that no error is encountered during the genotype to phenotype translation.

Program evolution with primitive grammar

This work appears in a paper entitled *"An Evolution of a Complete Program Using XML-based Grammar Definition"* (Siau et al. 2012). The initial work focused on adjusting the parameters experimenting with different crossover rates, mutation rates and population size. Further work to improve the technique is also included here, which proposes the use of a multi-objective fitness function. For consistency, every test was run many times and the results were recorded. The comparison of results for the best parameters setting are in agreement with Withall (2003); 7 genomes population with 10% mutation rate and 50% crossover appears to produce high fitness individuals quickly.

The initial population is generated with a random number sequence seeded by one of the first 10 prime numbers. A genome contains active genes and may also contain padding genes. An active gene will affect the phenome (solution) if it is changed (through the process of crossover or mutation). In contrast, a padding gene does not. A padding gene is an unused gene to fill up the block so that the same size of each block can be maintained in each

genome. Therefore, in a case where a particular program structure or statement requires fewer genes, these unused genes will be ignored. This should ensure that the next statement/structure translation would start from the first gene of the next block, thus its interpretation would also be unchanged.

A sample genome showing these two types of genes is depicted in Figure 4.10. The purpose of having these genes is to ensure the consistency of the Genotype-Phenotype translation as each block is independent of each other. A block of genes in a genome is translated to a line of code in the phenome, therefore, a ten-block genotype would produce ten lines of code phenotype.

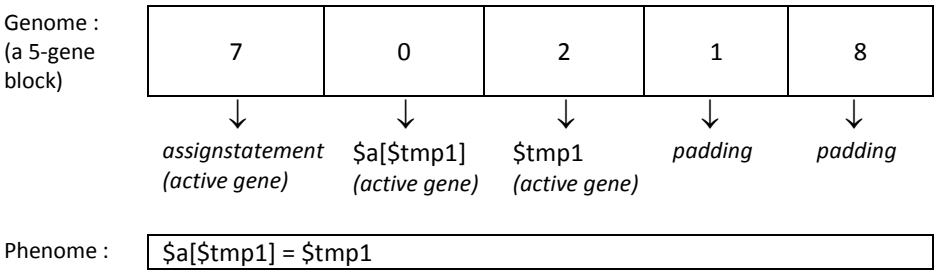


Figure 4.10 Active genes and padding genes in a block

Experiment and Results Discussion

The sorting list of integers experiment is designed to answer two questions. Firstly, how well the first algorithm, which avoids the 'repair function', performed in comparison to the previous two works. Secondly, to determine how the multi-objective fitness function affects the performance of the first algorithm.

The lines of code produced (program statements) are the result of the mapping of the genotype to its equivalent phenotype. The basic process of genotype to phenotype translation is by finding the remainder using a modulus operator. The same concept is applied here and the algorithm which describes the steps in Genetic Programming method is in Figure 4.11. Figure 4.12 depicts the Genome to Phenome translation process. Note that the first

integer of the first block always represents a statement and the assignment statement requires fewer genes, thus the unused genes are ignored.

Algorithm 1 Genetic Programming

Input: Set number of iterations, *iteration*, population size, *popSize*, the size of a chromosome, *chromosize*, the size of a block, *blockSize,* Rate of mutation, *MutationRate* , crossover probability, *Probability*, upper range of integers that a gene can take, *MaxRange*

Seeder: seeder
ListsOfIntegers: lists of integers to be sorted
Create initial population *pop* of *popsize*,
Apply Genotype-Phenotype-Mapping to *pop*
fit= fitness(*pop)*

for *loop1* = 1:*iteration*
 for *loop2* = 1:*popSize/2*
 Selects 2 individuals from *pop* based on fitness,
 Two offspring = crossover with Probability and mutate with *MutationRate*
 newpop += Two offspring
 Apply Genotype-Phenotype-Mapping to *newpop*
 newfit += fitness(*newpop*);
 end for
allpop = pop + newpop
allfit = fit + newfit
sort *allpop* in descending order based on their *allfit* value
pop = top *popsize allpop*
fit = top *popsize allfit*
end for
Output: *A program*

Figure 4.11 Pseudo-code of the Genetic Programming to evolve a program.

The produced program is considered useful if it achieved 100% fitness value and the survival of the genome depends on how fit it is in comparison to other genomes in the population. This means the seven fittest genomes will survive to the next process of reproduction out of the fourteen genomes in the population at each cycle. The less fit genomes will be discarded from the 'potential' population as they are assumed not viable to be processed further.

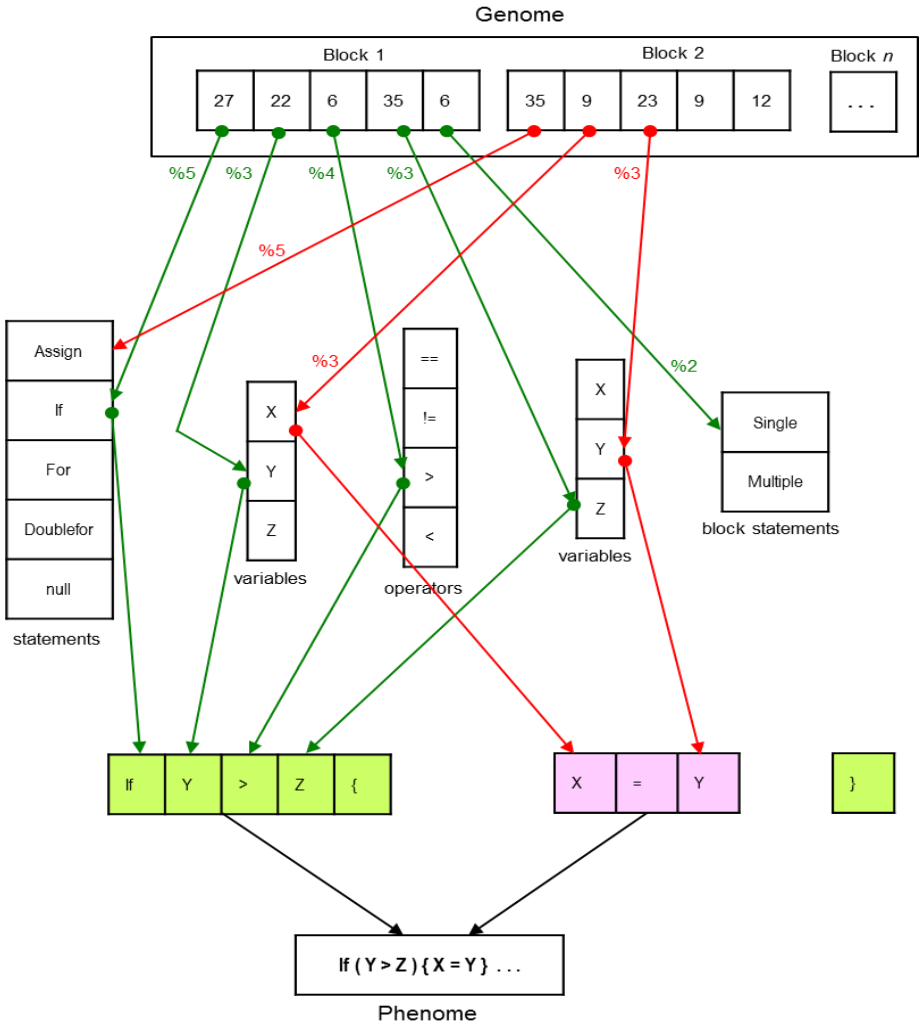


Figure 4.12 Genome to Phenome Mapping with a new grammar rule (block statements). This helps the mapper to decide if a particular statement (if, for or doublefor) has a single true statement or multiple true statements.

The results in Table 4.2 show the comparison of the four different approaches based on ten independent different seeding runs. The statistics generated from ten repeated runs is to ensure that one solution for the sort problem does not result in a 'lucky' fitness score based upon the starting population. 'E1' refers to Withall's approach, 'E2' refers to Xhemali's approach, 'E3' refers to a new approach introduced by this research (a replication of 'E1' approach with the clean grammar) and 'E4' is another approach introduced in this research (a replication of 'E3' with a slight change in the fitness function). The 'E4' fitness is measured using multiple objectives (a set of three weights is assigned to (a) the set of input integers in the lists, (b) the order of the integers and (c) the length of the lists). Unlike the other three approaches, the

maximum fit score in ‘E4’ is independent of how many integers in the test set, i.e. regardless of the length of the test set. It is important to note that seeding does not apply in ‘E2’ (Xhemali et al. 2010b).

Table 4.2. Comparison of results with the previous works (10 runs) – E1: Withall et. al. (2003), E2: Xhemali et. al.(2010b), E3: A replication of E1 with clean grammar, E4: A replication of E3 with a multi-objective fitness function.

	Genotype Length	Selection	Crossover	Mutation Rate	Generations		Std. Dev.	Mean	Median
					Worse case	Best case			
E1	Fixed length	Roulette Wheel	Uniform	10%	47975	1840	14837.19	21001.3	17442.5
E2	Variable length	Tournament	Uniform	One gene in each genome	35467	93	11796.03	10049.2	5320
E3	Fixed length	Roulette Wheel	Uniform	10%	36028	4407	11372.19	23982.6	27992
E4	Fixed length	Roulette Wheel	Uniform	10%	38068	543	11126.46	13004	9381

Given a whole set of different seed values, the number of generations and the time required by each run to find the solution are recorded and analysed. Furthermore, in order to ensure that the solution is valid and correct, the product (code) is dry-run manually and the output is analysed and compared (actual result and expected result). Figure 4.13 shows one of the complete and successful sort programs generated by the GP system (note the % indicates a modulus operator in PERL). Other evolved sort programs can be found in Appendix 4.

The standard deviation shows that ‘E4’ approach is the most consistent approach in finding a good solution. It can be seen that on average, ‘E3’ is the slowest. Note that ‘E3’ is a repeat of ‘E1’ with the improvement in terms of producing a syntactically correct program, independent of the repair function. The trade-off to this approach is the slow performance to achieve a fit solution. The mean is the highest among the four and the median is about 1.6

times greater than ‘E1’ and five times greater than ‘E2’. It was presumed that moving the grammar to an external file and an addition of a gene in the genotype required by the clean grammar contributed to this adverse performance.

```
for $tmp1(0..$#inlist){
  for $tmp2($tmp1+1..$#inlist){
    if ($tmp4>=$inlist[$tmp2%($#inlist+1)]){     #redundant
      $inlist[$tmp1%($#inlist+1)]=$tmp2;
    }
    $tmp3=$inlist[$tmp1%($#inlist+1)];
    if ($inlist[$tmp2%($#inlist+1)]<$tmp3){
      $inlist[$tmp1%($#inlist+1)]=$inlist[$tmp2%($#inlist+1)];
      if ($tmp2!=$tmp1){
        $inlist[$tmp2%($#inlist+1)]=$tmp3;
      }
    }
  }
}
```

Figure 4.13. A sample of an evolved Sort program generated by the evolutionary program.

Although Xhemali’s approach (‘E2’) is the fastest, the result is not consistent, as the initial population is created with random integers. This means that the experiments are not repeatable due to non-seeding of the random number stream. Because the result changes every time the system is run, this approach is highly dependent on ‘luck’ to get to the solution quickly (Xhemali 2010b). Furthermore, the GP used a single list as test data compared to 10 lists in Withall’s and this research.

Even though it difficult to replicate the results, the ‘E4’ approach is still within an acceptable distance from ‘E2’ with a median difference of 1.7 times. However, ‘E2’ has the same problem with Withall’s (E1), i.e., the use of repair function to produce the syntactically correct and complete program (refer to Section 3.2.4 for the issues of these two previous works). The fact that ‘E4’ consistently scores lower than ‘E1’ and ‘E3’ suggests that the performance improvement is attributable to the normalisation of the fitness function (length, size and set of the list).

Another attempt to look at the performance comparison between 'E1', 'E3', and 'E4' is by increasing the number of seeds to one hundred. Table 4.3 shows the performance of Withall's system and the new approach with respect to the generation and time taken to evolve the sorting program using one hundred random seeds. It was disappointing to discover that the new approach 'E3' is not as fast as Withall's. This is caused by the increase of the genome size and the search space for the 'clean grammar'. Based on all tests, 'E3' was only able to reach 91% fit-rate.

Generally, it will take longer to achieve a solution if the search space allowed by the representation is larger (Koza 1992). In this approach, the clean grammar requires an additional decision for determining the content of a block statement, i.e., whether it contains single statement or multiple statements, thus there is a doubling of the search space for a 1.6 increase in time. Also, this 'clean grammar' has introduced a properly structured set of rules validated using a DTD that produces error-free evolved program with no dependence to the 'repair function' and is external to the main program, which provides an easy access to other programs to read and append new rules.

Table 4.3. Comparison of 100 seeds results between Withall's works (E1), clean grammar (E3), a multi-objective fitness function (E4) based on generation cycle and time to achieve fit solutions.

	Generation			Time		
	E1	E3	E4	E1	E3	E4
N	100	100	100	100	100	100
Mean	15514.56	22906.53	11607.75	10.50	16.55	10.19
Std. Error-Mean	1081.17	1546.99	885.41	0.75	1.13	0.82
Median	12491	20527.50	10107	9.00	15.50	9.00
Std. Deviation	10811.69	15469.87	8854.10	7.52	11.31	10.19

Also, the results above suggest that an introduction of a 'multi-objective' fitness function has reduced the generations to half that of the 'E3' approach. This shows that normalising the fitness score for this problem has improved

the performance. ‘E4’ also exceeded the performance of ‘E1’, showing a decrease of 25% in the number of generations required to find the solution. However, the standard deviation for ‘E4’ is about 0.27 times higher than ‘E1’. This is due to more of the unsuccessful evolved programs reaching the allowed time of execution.

One interesting fact experienced from this evolution is the built-in function provided by the chosen programming language library, specifically the use of rand() to generate a random number. In the case of this experiment, the integer for the genes in the genomes is generated randomly using the rand() function. It was found that the generated random numbers in Windows environment of different Operating System (32-bit and 64-bit) are not identical, as well as those generated by MAC OSX. This means the results produced by the same program in these different environments are dissimilar and hence care must be taken to ensure their integrity and consistency, especially important when doing comparisons.

4.4.3 Sorting Lists of Integers (Seeded)

In Genetic Programming, the success of an encoded solution to solve a problem is reliant upon the set of parameters or operators relative to the initial population, type and probability of crossover, type and probability of mutation, termination criteria, the parent selection methods and fitness evaluation function. This experiment finds the effect of manipulating the initial population on the above result by biasing the gene values for it to start with.

Initial populations can be created in three ways, the most common method is where individuals are generated randomly using a pseudo-random number generator. This provides great variation amongst individuals and keeps the population mostly unbiased. Seeding the Initial population with known solutions to the problem domain with an attempt to improve them is another method. One drawback is that this way is likely to confine the search to one area, making it harder for the GP to find better solutions. In the third method,

the population is randomly created using predefined blocks of genes, which provides some initial knowledge of the problem domain.

Experiment and Results Discussion

The experiment in this section concerns the second method, i.e. seeding with a known solution. Two approaches are introduced and compared. The effect that these techniques would have on the performance of a GP is assessed. The GP is configured the same way as before to carry forward seven individuals to the next generation using 50% crossover probability and 10% mutation rate. The first technique is to have the first chromosome in the population seeded from existing solutions and the rest of the population are created at random. In this thesis, this technique is referred to as 'Solution Seeded'. It provides a chromosome of a good solution to start with without putting much restriction on the solution area it can search.

The other technique involves taking an existing solution and applying 'one gene per chromosome' mutation, referred to as 'Solution Mutated'. This technique assumes that the solution to the problem is not far from the existing one. This could be just a change of a single statement type or a variable name.

Table 4.4 shows the statistical result of the first technique – 'Solution Seeded' (first chromosome seeded from existing solution) in comparison to the second technique – 'Solution Mutated' (one chromosome randomly mutated). The complete set of results can be found in Appendix 5. Each experiment is run 100 times with a different random seeds based on 100 prime numbers.

Table 4.4. Summary statistics of the generations required to find a solution (100 runs) for Solution Seeded and Solution Mutated.

Method	Median	Max	Mean	Std. Dev.
Solution Seeded	156	1206	256.67	304.262
Solution Mutated	312	3346	454.43	471.188

The results were calculated using IBM SPSS 20.0 and evaluated using Paired T-Test. The data provided sufficient evidence that Solution Seeded produces fit solutions in fewer generations than Solution Mutated with a confidence level of 95% for this kind of problem. The standard deviation indicates that Sort Seeded is also the most consistent approach in finding a good solution. Furthermore, the small standard deviation for both Solution Seeded and Solution Mutated indicates that they can sometimes become trapped in local optima.

4.4.4 Reverse-Sort Lists of Integers (Seeded)

In human programming, a large program can be broken down into a main program and a set of parameterised functions, in which these functions can be invoked repeatedly. The breaking down of big program into smaller and manageable subprograms is called modularisation. In practice, modularisation can be seen as separating partial solutions into independent modules that each solves one aspect of the sub-problem.

The same concept has been successfully adopted in Evolutionary Computation, such as Genetic Programming (Koza 1994) and Grammatical Evolution (Harper & Blair 2006); with a goal to increase the scalability and complexity of the problem it can solve. If only core and rigid grammar rules are being used, this means that only a limited set of problems with simple specifications can be solved (Koza 1990; Withall 2003). However, one of the well-known issues of evolutionary computation is the search space; if it is larger, it can cause a negative impact on the evolutionary performance.

The next experiment is using the same GP method as above to evolve a reverse-sort program that sorts the lists of integers in descending order with the grammar rules slightly altered. Not only does the grammar contain primitive rules, but it also contains subprograms. Subprograms are the previously evolved programs, which are appended to the original grammar as a function. They are offered as a component of one of the statements in the

grammar called ‘functioncall’ and each has a specific task that it solves. Similar to the ‘functions’ in modularised programming, these evolved functions may be used to solve partial problems of a larger problem in the future. For example, a swap function, which exchanges the value between two variables, is useful for a sort program (main program) to switch the positions of data if they are in the wrong order. This method allows extensions to the grammar to solve more complicated problems.

In addition to the above GP method, additional scripts are added to automatically add the successfully evolved program back into the original grammar as a function and update the ‘functioncall’ rule component. The name of the function will be taken from the name of the file being evolved. It is important to note that the function is only considered valid if its fitness score is 100% and no duplication is allowed within the grammar file. Moreover, these functions may be inserted zero or more times within a generated program. However, similar to the issue which arose above, additional overhead is required to read and write functions to the XML file. Nevertheless, this is minimal.

Experiment and Results Discussion

An experiment using the extended version of the grammar in Figure 4.14 has been carried out to discover the feasibility of this extension and examining its impact to the evolutionary program. The number of the various functions in this grammar is done in an incremental way and the grammar used in the final test has all the three functions (i.e. swap, swapMin and swapMax), each accepts two integer parameters. The program uses the same ten lists of integers to ensure repeatable results.

A function can be viewed as highly useful, less useful or least useful to the problem to solve. In this reverse-sort problem, ‘swapMax’ function is considered very useful as it produces a descending order of two integers, ‘swap’ function is less useful as it only swaps two integers without the need to know if one is bigger than the other or otherwise, ‘swapMin’ is the least useful

as it arranges two given integers in ascending order.

statementseq	*::=*	*statement | statements*
statements	*::=*	*statement statementseq*
statement	*::=*	*nullstatement | assignstatement | ifstatement | forstatement | nestedforstatement | functioncall*
nullstatement	*::=*	*";"*
assignstatement	*::=*	*wvar "=" rvar*
ifstatement	*::=*	*"If" "(" wvar opr wvar "{" statementseq "}"*
forstatement	*::=*	*"for" "(" cntr "=" 0 ";" "N1" "<" length ";" "N1" "++" ")" "{" statementseq "}"*
nestedforstatement	*::=*	*"for" "(" cntr "=" 0 ";" "N1" "<" length ";" "N1" "++" ")" "{" "for" "(" cntr "=" "N2 + 1" ";" "N2" "<" length ";" "N2" "++" ")" "{" statementseq "}" "}"*
wvar	*::=*	*"a[tmp1]" | "a[tmp2]" | "tmp3" | "tmp4"*
rvar	*::=*	*"a[tmp1]" | "a[tmp2]" | "tmp1" | "tmp2" | "tmp3" | "tmp4"*
op	*::=*	*"==" | "!=" | ">" | "<" | ">=" | "<="*
Length	*::=*	*"length"*
cntr	*::=*	*"tmp1" | "tmp2"*
functioncall	*::=*	*swap | swapMax | swapMin*
swap	*::=*	*swapNum "(" rvar "," rvar ")"*
swapMax	*::=*	*swapMax "(" rvar "," rvar ")"*
swapMin	*::=*	*swapMin "(" rvar "," rvar ")"*

Figure 4.14. The extended version of the program statements syntax expressed in BNF.

Table 4.5. A comparison between the reverse-sort program and the optimised reverse-sort program using evolved functions (measured in terms of the number of required generation).

Seed	No function	1 function			2 functions			3 functions (swap, swapmax, swapmin)
		swap	swapMin	swapMax	swap, swapMin	swap, swapMax	swapMin, swapMax	
1	9114	52	0	21	67	0	40	0
2	4407	875	11	11	38	11	27	48
3	27830	473	96	16	60	27	66	75
5	36028	668	5	2	42	4	6	34
7	24400	135	17	17	6	11	98	74
11	31384	1334	40	2	83	97	66	56
13	31190	313	20	35	41	30	20	57
17	11928	222	5	13	54	19	2	7
19	35391	542	14	51	179	14	9	75
23	28154	749	31	9	127	90	40	40
Mean	23982.6	536.3	24	17.7	69.7	30.3	37.4	46.6
SD	11372.19	389	28.13	15.12	49.9	34.6	31.23	26.88
Median	27992	507.5	15.5	14.5	57	16.5	33.5	52

The aim of the evolved program was to arrange the lists of integers in descending order (reverse-sort). There are eight different grammar contents being tested; one test with no function, three tests are made with one function each, three tests are made with pairs of functions (one is highly useful and the other is less useful) and finally, one test is made with all three functions. These variations are to compare how they influenced the performance of the GP system. The result in Table 4.5 shows comparison between the performances of the GP system. It was observed that if useful functions are defined, it was helpful to achieve an optimised behaviour; in particular, it has reduced the fitness evaluation. The worst case is evolving reverse-sort with primitive rules alone (indicated by 'no function').

These experiments using a 'clean' grammar show that adopting the concept of modularisation to solve a computer problem improves the effectiveness of the GP. The optimum solution can be reached within a few generations. It is important to note that the required generation time increases if the number of functions increases – the search space is larger. Figure 4.15 shows a sample of the evolved reverse-sort program using the grammar with a swap function. The general work from this software evolution is then applied to the evolution of regular expressions, which are used to match the required information from the relevant web pages.

```
for $tmp2(0..$#inlist){                    # loop until the size of inlist is reached
   ($inlist[$tmp2%($#inlist+1)],$inlist[$tmp2%($#inlist+1)])=
     &swap($inlist[$tmp2%($#inlist+1)],$inlist[$tmp2%($#inlist+1)]);   #function call
}

for $tmp2(0..$#inlist){
   for $tmp1(0..$#inlist){
      if ($inlist[$tmp2%($#inlist+1)]>$inlist[$tmp1%($#inlist+1)]){
         ($inlist[$tmp1%($#inlist+1)],$inlist[$tmp2%($#inlist+1)])=
           &swap($inlist[$tmp1%($#inlist+1)],$inlist[$tmp2%($#inlist+1)]);
       }
    }
}
```

Figure 4.15. PERL: An Example solution for a *swap* function embedded in a Reverse-sort program generated by the evolutionary program. Note that the first loop is redundant and the second nested loops form a simple bubble sort.

4.4.5 Distance from Mean

The ‘DistanceFromMean’ experiment is designed to answer two questions. First, how well the ‘clean’ grammar helps to construct a program, which requires a sequence of block statements to solve a particular problem. Second, how well the GP system finds a good solution.

Given a list of integers, the ‘DistanceFromMean’ finds the distance of each of these integers from their mean. To achieve the correct result, two loops are needed. The first loop is to calculate the sum of the numbers in the list in order to find the mean, while the second loop is to calculate the distance from the mean. The specification is in Figure 4.16. It simply says that the input is a list of integers and the output is a list of real numbers. The length of the input should be equal to the length of output and *i*th element of the output list is the difference between the *i*th element of the input list and the mean of the input list. The mean is calculated by dividing the sum of the input integers by the length of the input.

$$
\begin{array}{l}
DistanceFromMean : \mathbb{Z}* \rightarrow \mathbb{R}* \\
pre\text{-}DistanceFromMean(Lin) \triangleq True \\
post\text{-}DistanceFromMean(Lin,Lout) \triangleq \#Lout = \#Lin \wedge \\
\qquad \forall i,\ 0 \leq i < \#Lin \Rightarrow Lout[i] = Lin[i] - Average(Lin) \\
where\ post\text{-}Average(Lin,A) \triangleq A = Sum(Lin)/\#Lin
\end{array}
$$

Figure 4.16. Specification for *DistanceFromMean* problem.

The fitness function is in Figure 4.17. The fitness was calculated based on the closeness to the correct distance, i.e., using absolute difference between the expected and the actual results. This approach meant that fit solution has zero fitness score and the larger the score, the worse the individual. Thus this was achieved with normalisation by subtracting the score from a maximum value of 5000. This technique is different from the previous method of incrementing the fitness value if the values are equal. Unlike the previous fitness technique where the individuals only improved to a certain point before finally stuck to a certain point, the later test, i.e., using absolute difference,

showed that the DistanceFromMean evolved much more easily. This is assumed to be caused by this particular specification having a small test.

```
$sum = 0;
foreach my $x (@L) {
  $sum += $x;
}
$ave =$sum/$#L;
for my $d(0..$#L) {
 $dist[$d]= abs($L[$d]-$ave);
}
 if($#results > 0)     {
       for my $n (0..$#results)     {
         $fitness += abs($results[$n]-$dist [$n]);
       }
 }
```

Figure 4.17. Fitness Function for *DistanceFromMean* problem.

The grammar to guide the generation of the DistanceFromMean program is in Figure 4.18. The same set of statements are applied except for some additional statements (add, divide and subtract), which are essential to calculate the sum, average and the difference from the mean. The 'double for' statement was removed as it does not provide any benefit to the evolutionary process, other than increasing the time required to reach the fit solution. An unnecessary statement ('Multiply') to solve the problem is also added to see if any interesting use was made of it by the evolution. However, the fit programs

statementseq	*::=*	*statement | statements*
statements	*::=*	*statement statementseq*
statement	*::=*	*nullstatement | assignstatement | ifstatement | forstatement | add | subtract | multiply | divide*
nullstatement	*::=*	*";"*
assignstatement	*::=*	*wvar "=" rvar*
ifstatement	*::=*	*"If" "(" wvar opr wvar "{" statementseq "}"*
forstatement	*::=*	*"for" "(" cntr "=" 0 ";" "N1" "<" length ";" "N1" "++" ")" "{" statementseq "}"*
add	*::=*	*wvar "=" rvar "+" rvar*
subtract	*::=*	*wvar "=" rvar " –" rvar*
multiply	*::=*	*wvar "=" rvar "*" rvar*
divide	*::=*	*wvar "=" rvar "/" rvar "if" "(" "N2" "!=" 0 ")"*
wvar	*::=*	*"dist[tmp]" | "ave"*
rvar	*::=*	*"$inlist[$tmp%($#inlist+1)]" | "inlist[tmp]" | "ave" | " length"*
op	*::=*	*"==" | "!=" | ">" | "<" | ">=" | "<="*
length	*::=*	*"length"*
cntr	*::=*	*"tmp"*

Figure 4.18. Program statements syntax expressed in BNF for the DistanceFromMean problem.

showed that this statement was made redundant. The ‘divide’ has a special control *if statement* embedded. Note that the <N2> in the grammar corresponds to the second non-terminal in the statement. The control is essential to ensure division by zero is avoided that could interrupt the execution of the program.

Experiment and Results Discussion

Unlike the previous experiments, the approach was tested using only a single list of integers *$inlist = {90,30,50,60,80}*. Because the ‘clean’ grammar was designed to avoid forcing the evolutionary program to add the required close brackets at the end of the program, a sequence of structured statements (for statement or if statement blocks) is achievable.

Figure 4.19 shows an example of a fully fit solution. It is noted that a tidying-up script could be put in place to remove all the null statements (‘;’). The null statement is commonly used as a placeholder in loop statements or if statement and it requires no action. It is included in the grammar as the solution program may require shorter ‘performing’ codes. This tidying is beyond the concern of this thesis as having the null statement does not affect the execution or the result of the evolved program.

```
$dist[$tmp]=$ave/$test[$tmp] if ($test[$tmp]!= 0);
;
$dist[$tmp]=$test[$tmp];
for $tmp(0..$#test){
   $ave=$ave+$test[$tmp%($#test+1)];
   ;
   ;
}
$ave=$ave/$size if ($size!= 0);
for $tmp(0..$#test){
   $dist[$tmp]= abs($test[$tmp%($#test+1)] - $ave);
}
return @dist;
```

Figure 4.19. An example of successfully evolved *DistanceFromMean* program incorporating a sequence of two loops.

Table 4.6. The results of the *DistanceFromMean*.

#	Generation
1	47141
2	21319
3	16220
4	38023
5	18174
6	10181
7	2192
8	35182
9	13094
10	3329

The result of ten runs of the experiment is in Table 4.6. This shows that the approach proposed in this research has successfully demonstrated that it is possible to solve a problem, which requires the use of a sequence of loops using any standard language, by applying the clean grammar proposed in this thesis. This experiment provides evidence that a 'repair function' can be avoided, without the need to include any specific construct in the grammar, like 'double for' in the previous experiments. Using a similar grammar definition, Withall's method (Withall 2003) failed to achieve this sequence of conditional statements structure due to separation of an end statement ('}') from the main conditional statement. This means his method could just as easily generate all the end statement at the end of the evolved program.

4.4.6 Results Summary

In conclusion, two methods have been highlighted and tested to deal with issues of evolving solution (program). Applying a fixed block genome ensures that the generated program is complete without reusing or creating new genes, while applying a 'clean grammar' (proper subsets of programming language) ensure that the evolved program follows the correct programming language syntax, which can cause problems during the fitness evaluation if not handled properly. The experiments, which were based on the five approaches (i.e. 'random' population, 'Solution Seeded' population, 'Solution

Mutated' population, multi-objective and modularisation) to evolve the computer programs, have shown that GP with modularisation demonstrated a great improvement on the system performance.

4.5 Chapter Summary

This chapter outlines the development and evaluation of software evolution system. The software evolution aims to generate a solution to the 'sorting' program, 'reverse-sort' program and 'DistanceFromMean' program. The experiments demonstrated the kind of programs which involve only a single program and a program calling previously evolved functions to solve a variety of problem specification.

Results show that the CoPE technique introduced here is not as fast as the previous work and evidence of a doubling of the search space for a 1.6 increase in time. However, this technique has provided better grammar structure that can be used to support the generation of an error-free program without depending on a 'repair function'. Further experiments were conducted and it can be seen that huge improvements on the performance to find the required solution have been achieved by adding seeded initial population, modularisation concept and a multi-objective fitness function. Furthermore it was shown that the new technique was able to evolve programs that could not be evolved by the older techniques that relied on the repair function, an example of this is the 'DistanceFromMean' program.

The next chapter concerns the evolution of the regular expression, referred to as REGEXEV. Some general features relevant to this evolution system are provided before the REGEXEV experiments are discussed in detail.

Chapter 5

Practical Application of GP – Domain 2

5.1 Chapter Overview

Chapter 4 described an evolution of a complete program to find useful programs to solve ‘sorting’, ‘reverse-sort’ and ‘DistanceFromMean’ problems. The method which incorporates modularisation demonstrated the best result. This chapter provides the various experiments aiming to apply similar approach to another domain, in particular, the regular expressions.

The next section introduces the experiment set up, the result of analysis based on 45 websites containing 80,950 web pages to find how the course attributes are represented on the relevant web pages, discussion on the regular expressions which significantly identifies the relevant course attributes and the experiments using live web pages. The fitness of the evolved regular expressions is important to determine these regular expressions usefulness to extract the required data. Therefore, a similar approach as the ‘COPE’ (Section 4.4.2) is applied which require the maximum fitness score to be reached before the evolved regular expressions are ‘fully fit’ to be used by the system.

A comparison study is also conducted to contrast the work in this research with Bartoli et al. (2008), which sees the use of GP and grammar to evolve regular expression to identify and capture the required data. Bartoli et al. reported the result of their approach to extract the title and the phone number from Wikipedia and W3C websites. However, the comparison reported here in Section 5.3 only applies to the title extraction as this matches the course attribute (title of courses), which is the focus of this research.

5.2 A Regular Expression Evolution (REGEXEV)

5.2.1 Introduction

Aims of Experiment

The REGular EXpression EVolution (REGEXEV) experiments are to explore how GP methods applied in CoPE (in Section 4.3.2) can be extended to different domains, specifically regular expressions. This initial stage looked into the feasibility of using Genetic Programming guided by structured extraction rules to evolve regular expressions for data extraction from web sources. Another aim was to be able to provide a set of criteria for the fitness function to improve the quality of extraction.

Subject

The evolution method concerned the generation of a regular expression from a regular expression grammar similar to generating a program from a program grammar (CoPE). The new evolved regular expression pattern should be useful for the information extraction to capture the data on a particular 'never seen before' HTML web page. The experiment was limited to single page extraction activity, where a single record or multiple records are listed on a single web page.

Specific Setting

i. The population size was 10.
ii. REGEXEV was designed to extract 'single record' or 'multiple records' from a single web page.
iii. The web pages were relevant to the training course domain in the UK.

5.2.2 Web page structure of the Training courses website

In the investigation, it was found that the websites of training courses usually render course information in a semi-structured manner. Details of courses offered are commonly presented in tables, lists or free text or a mixture of these. Thus, this closely meets the criteria described by the semi-structured data in Section 2.1. Frequently, different formats are used to denote the same concept. Consider the following samples of training information:

a. Course name: Introduction to JavaScript Programming
Date commencing: 12/10/2010
Duration: 5 days
Fee: £697 + VAT
Location: Nottingham University

b.

Title	Introduction to JavaScript Programming
Start Date	12 Oct 2010
End Date	17 Oct 2010
Location	Nottingham University
Price	£697

c.

Course Title	Location	Date	Duration	Price
Introduction to JavaScript Programming	Nottingham University	12-10-2010	5	697
Effective Presentation Style	Leicester	13-12-2010	3	700

d. Price: £697 + VAT
Location: Nottingham University, Beeston, Nottinghamshire NG7 2RD

Title	Duration (Days)	Oct	Nov	Dec
Introduction to JavaScript Programming	5	12		
Stress Management	3		23	

Figure 5.1 (a-d) Sample information presentation styles.

The above examples (Figure 5.1) show the different styles commonly used to present information on the web page. We can interpret them as each sample containing information about a course; (i) title of course (ii) location (iii) start date and (iv) price of the course. However, the same interpretation is complicated for a computer program to emulate. A similar outcome to human capability can only be competently achieved with the aid of data patterns or rules with a provision that these patterns or rules are extensible. This could

possibly mean that some human assistance needed to be sought in order for the computer system to learn and discover new data presentation styles.

A pattern is defined by NIST (2005) as an expression of a specific form that is used for matching text during the extraction process. Patterns are normally created for content and context of data. Because web pages are designed and presented for human view, the underlying structures for presenting the information vary from one web page to another. For example, the term “title of course” can be written about 20 different ways. The pattern analysis aims to find the similarity and the differences of data patterns/structure from the web pages of training courses. Some of the web pages have a different layout although they belong to the same website.

Although these information presentation styles are not uniformly formatted, there are still some similarities in the information presented. For example, most price values start with symbol £, some digits and sometimes with a dot and two more digits for the pence denomination and they are explicitly labelled with keywords such as ‘price’, ‘fee’ or ‘cost’. It is through the discovery of these conjoint presentations of information that data pattern/rules can be framed and used for extraction.

As stated earlier in Chapter 1, the REGEXEV approach is aimed to extract course title, location, date of the course and the cost. Note that in some cases, some of this information is not presented on web pages. Some of the reasons are:

1. Although the course is listed on the web page, the course is not yet offered in the near future; therefore the date is not available.
2. The trainer normally uses the requesting organisation’s site to conduct the course, so the location is not specified.
3. The cost is not presented, as the trainer offers tailored courses according to the organisation’s needs.

Table 5.1 shows the proportion of the data of interest presented in 45 websites containing 80,950 web pages. Although, there are various ways that

can be used to display the information, exploration has shown that the majority of web pages in this domain use a table. The cost of the courses is commonly presented with the pound (£) symbol with or without the pence (Table 5.2) whereas the two popular ways the title is presented are using font size formatting, that is, <H*i* > where *i* represents a number from 1 to 3 (with total of 40,214 web pages) and 'no format' (Table 5.3).

Table 5.1. The number of web pages showing how the specific data is displayed.

	table	Division	Paragraph	List	Not Available
Date	**52,823**	24,086	253	3339	449
Cost	**53,169**	551	26,810	317	103
Location	**53,361**	453	26,869	177	90
Title	**41,614**	39,237	1	25	73

Table 5.2. Price representation on web pages.

	£	GBP	£ and GBP	Not available
No. of web pages	**74,698 (92%)**	6,070	79	103

Table 5.3. Title of Course representation on web pages.

	No format	<H1>	<H2>	<H3>	<B>	IMG	none
No of web pages	**40,306**	15,797	24,148	269	303	54	73

Dates and prices are structured word objects, which are strongly structured (Baumann et al. 1995). This means that a price, for example, contains numbers headed or followed by some classifying unit (currency symbol). Therefore, they can be treated with predefined constraints. In contrast, the titles as a segment of text are quite difficult to recognise (Hu et al. 2005). Nevertheless, title page or header of a document is a strongly structured document part (Baumann et al. 1995). Hu et al. (2005) stated that although <title> tag explicitly specifies the title field of the web page, this is not practically included in all web pages and about 33% of the title fields are bogus. This is because titles in the HTML body are more obvious to readers compared to title fields embedded in the meta-title. Recognising locations also has the same difficulty.

The input from the analysis of the patterns in web pages was used to

construct the knowledge model (keywords) and to design the valid combination of the DOM tree components and the correct patterns in regular expression notations. This provides knowledge on how the algorithm should be designed for the extraction task. The next section describes the data pattern (regular expressions) method.

5.2.3 Regular Expression to define web data pattern

One of the merits of regular expression is that it is a powerful tool to discover diverse patterns of textual strings. Because regular expression has high complexity in both syntax and grammatical rules compared to computer program, building a good regular expression to match a data pattern can be extremely challenging for a human, although some data such as price (containing a £ symbol, digits and the pence) might only require simple expression, for example, *£[0-9]{1,4}(.[0-9]{2})?,* which means that a price has a pound symbol followed by 1 to 4 digits, each ranging from 0 to 9, followed by an optional dot and 2 digits ranging from 0 to 9.

In this research, regular expressions are evolved to discover and extract the attributes of course/courses (appearing in single record or multiple records), which are the title, date, price and location. This extraction is only applied on a single training course, a web page at a time. However, a regular expression may be useful on one web page but useless on another. Thus, automatically evolving the regular expressions would save a considerable amount of human time and effort compared with handcrafting the variety of regular expressions to meet different requirements.

Using a similar method to the CoPE system above, a common template for the GP is constructed for all the course attributes. For each, the addition of the main body of the fitness evaluation function is required. This is because each attribute of interest would require a specific fitness function and specific data format pertaining to this attribute. Nevertheless, they use the same grammar rules to locate the relevant attribute on the web page. Similar to

CoPE, an additional script was also coded that allows the recording of the experiments' results in a text file, based on random number seeds for future analysis.

Table 5.4 shows some examples of the regular expressions that could be used to match the course attributes format, which the GP aims to achieve. It can be observed that Text matcher will match any text that fits the pattern while DOM matcher will capture anything in between the specified tags. The Text Matcher or DOM Matcher may match irrelevant data, if individually applied. Using a joint approach of Text Matcher and DOM Matcher makes it possible to optimise the extraction performance. However, it is important to have a good fitness function to decide between these two techniques which matcher is superior to the other. This issue is further discussed in the following sub-section (Fitness Test).

Table 5.4. Regular Expressions to define the patterns of the specific piece of data.

Elements	Regular Expression Pattern	Example Matches
(1) Text Matcher		
price	£\d+(\.\d{2})?	£210.00
date	\d{1,2}[\/.-]\d{1,2}[\/.-]\d{1,4}	20/10/2011 *or* 20-10-2011 or 20.10.2011
Location	(\w\s?)+	Birmingham City *or* Course Fee
Title	(\w\s?)+	Javascript Programming *or* Course Fee
(2) DOM Matcher		
A table cell (td)	<td[^>]*>(.*?)</td>	<td>Course Name</td> *or* <td id="row2">£210.00</td>
A division/section of a web page (div)	<div[^>]*>(.*?)	Price : £210 *or* Cost
A paragraph (p)	<p[^>]*>(.*?)( )?	Location: Birmingham
(3) Text and DOM Matcher		
price	<td[^>]*> (£\d+(\.\d{2})?) </td>	<td>£210.00</td>
date	<div[^>]*>(\d{1,2}[\/.-]\d{1,2}[\/.-]\d{1,4})	20/10/2011
location	<p[^>]*> ((\w\s?)+)	Location: Birmingham

In this research, a clean grammar is used to guide the system to build a valid regular expression pattern automatically to capture the relevant information on the web. The clean grammar concept containing the extraction rules is described in the 'Clean Grammar' sub-section below but first, the fitness function for this domain evolution is presented.

Fitness Function

This section describes the fitness function to validate the efficiency of the evolved regular expressions to match the title, date, price and location of a course on a web page. Unlike the programming domain which calculates the fitness test based on formal specification, evaluating the 'unknown' output from regular expression is not easy. This is because the regular expression may extract false information. Some test criteria should be tested against the extracted information and the weighted sum of all the individual criteria values is then used to calculate the total score for each regular expression. The test criteria chosen should be general so as to be capable of working with the training course web pages.

In this research, a unique set of criteria is proposed for each of the course attributes. The criteria reflect the various kind of features based on words (token) in a text and structural information of each attribute. The criteria were carefully chosen based on the analysis of the 80,950 web pages from 45 websites. The fitness function is constructed by testing the output of the GP system with this set of criteria. The fitness of each test could be weighted to give one test more impact on the overall fitness. However, all the fitness increments are identical in this thesis. This is because that these criteria are equally important to determine the correctness of the extracted data. Each criterion contributes a point towards the fitness score if it is satisfied. A score of one represent true match and a zero otherwise. The overall fitness value for each attribute is the sum of all the fulfilled criteria.

Each course attribute is given its own fitness function. Although some of the criteria are common for all course attributes, it is assumed that each attribute

would have some distinguishable criteria such as the data pattern and 'Relevance' and 'low-relevance' corpuses, thus the reason for encoding individual fitness functions. 'Relevance' corpus consists of the desired words that are related to the course attribute to be matched e.g. VAT is relevant to a price, while 'low-relevance' contains undesirable (negative) words in the web page that are not related to the course attribute e.g. postage is not relevant to the course price. The list of relevance and low-relevance corpuses is in Appendix 2. These lists in the current prototype (including the keywords list) are collective words manually created after a thorough analysis of the relevant webpages mentioned above and they are not exhaustive lists. Choosing the correct words involves the domain expert to manually analyse the extracted information and update these lists if new words exist. In future, these lists can be automatically updated through the provision of positive and negative example and the extracted information stored in the TS-WIE database (described in Chapter 6). Table 5.5 below summarised the fitness criteria for each course attribute and the following subsections describe the criteria relevant to each course attribute.

Table 5.5. A Summary of fitness criteria applicable to the course attributes.

Criteria	Title	Location	Date	Price
Regular Expression validation	✓	✓	✓	✓
Title field & similarity check	✓			
Text pattern e.g. [A-Za-z0-9]+	✓	✓	✓	✓
Tags validation	✓	✓	✓	✓
length validation	✓	✓	✓	✓
Word's length validation	✓	✓		
Digit validation			✓	✓
Validate against stored location in the database	✓	✓		
Relevance and low-relevance corpuses validation	✓	✓	✓	✓
Expressed after keyword		✓	✓	✓

Course title

Existing work on title extraction concentrates mainly on online research papers, such as Hu et al. (2005) and Zhang et al. (2005). Fortunately, these kinds of paper are well-formed documents; therefore some constraints can be

set to isolate the relevant section, thus increasing extraction accuracy. For example, titles are normally placed on the top part of the document, followed by the author names, affiliation and abstract. There is also a distinction between its features than the rest of the document, for example, generally the font size is bigger and bolded. On the contrary, titles of courses on web pages are rarely at the top, especially when the page contains multiple courses on offer.

The two methods employed by Xue et al. (2007); DOM tree feature and Lexical feature are also applied here. An evolved regular expression pattern is given the maximum fitness score if HTML tag of the extracted data matches a large header (<H1> or <H2>) and if the extracted data matches at least 80% of the words in this title field. If this is not the situation, then the fitness score is the sum of all the criteria; 1 if a criterion is met and 0 otherwise.

The regular expression fitness algorithm is expressed as follows:
When the evolved regular expression pattern is applied to the page the string that it matches is called the extracted data. The pattern is given a score as follows:

If the HTML tag of the extracted data is a large header and the extracted data matches at least 80% of the words in the title field of the page
Then

fitnessScore = maximum fitness score

Else

fitnessScore is equal to the number of successful criteria from the following

i) valid regular expression pattern, ii) text pattern, iii) tags

iv) number of words, v) word length, vi) not a location, vii) word lists.

The criteria in more detail:

- Valid regular expression pattern: The regular expression is evaluated and a fitness score of 1 is given if it is a valid expression.

- Text pattern: The output is compared with some pre-specified text pattern from the database such as *([A-Z][a-z]+\s*)+* and *([A-Z][a-z]+\s*)+([A-Za-z0-9]+\s*)+*. Criterion is met if the output matches the pattern.
- Tags validation: The criterion is fulfilled if the tag of the extracted data is a header.
- Number of words: The length of the extracted attribute is used to check if it is a valid title. Based on the analysis of the titles from the 45 websites, the maximum number of words for a title is 20 words or less. If this is satisfied then the fitness is incremented otherwise its value remains the same.
- Word length: This criterion is satisfied if every word is 20 characters or less in length. This maximum length is determined by the longest word in the title from the sample web pages (randomly taken from the 45 websites).
- Validate against stored location: Title and Location share the same textual cue, thus checking against stored locations (town, cities or countries in the UK) avoids capturing a location. The fitness remains the same if the stored location equates the extracted data or if it appears as a substring of the extracted data.
- Relevance and low-relevance words lists: At this moment, the relevance and low relevance corpuses are already stored in the table, which are manually created after a thorough analysis of the relevant web pages. This could be made updatable through provision of training examples in a semi-automatic approach. The method compares the extracted data against the common and negative terms used specific to the course title. If the data matched the 'relevance' items, then the criterion is said to be satisfied, thus the fitness is incremented. Unlike the 'low-relevance', if the data does not match the list of words in this corpus, the fitness increases.

Course Location

Designing the algorithm to distinguish the location from the title is difficult as both have the same lexical features and valid length. However, the last three criteria help to make this location distinctive.

The regular expression fitness algorithm is expressed as follows:

If the tag of the extracted data not equals to <title>

Then

fitnessScore is equal to the number of successful criteria from i) valid regular expression pattern, ii) text pattern, iii) tags, iv) number of words, v) word length, vi) matches stored location, vii) word lists, viii) after keyword.

The details of the criteria are below:

- Valid regular expression pattern: The regular expression is evaluated and a fitness score of 1 is given if it is a valid expression.
- Text pattern: the example of valid data patterns are *([A-Z][a-z]+\s*)+* and *([A-Z][a-z]+\s*)+[A-Za-z0-9]+\.?\s**
- Tag validation: It is uncommon that the location appears in the large heading. This criterion is satisfied if the tag of the extracted data does not match <H1> or <H2>.
- Number of words: The fitness is incremented if the extracted string contains 10 words or less, or if it is part of a long sentence/paragraph, the keyword should be found in front of it.
- Word length: This criterion is satisfied if the length of each extracted word is 20 or less characters.
- Validate against stored location: A list of locations in the UK is stored in the database. The extracted data is valid if it matches any one of the locations either wholly or partly. It is worthwhile keeping this list as it does not require frequent update (addition of new location is quite rare as it usually exists).
- Relevance and low-relevance words lists: These are lists of positive

and negative words respectively to eliminate extracting normal text.

- Expressed after keyword: This criteria is useful to identify a location, which is not available in the database. Common keywords used are location, venue and place.

Course Date

Date and Price are structured word objects (Baumann et al. 1995). They are strongly structured objects, which can be described by a formal method, such as grammar and regular language.

The regular expression fitness algorithm is expressed as follows:
fitnessScore is equal to the number of successful criteria from the following
i) valid regular expression pattern, ii) text pattern, iii) tags, iv) number of words, v) digit validation, vi) word lists, vii) after keyword

The criteria applicable to determine its fitness are described below:

- Valid regular expression pattern: a fitness score of 1 is given if the generated regular expression is a valid expression.
- Text pattern: This criterion ensures that the extracted text is a valid date by comparing it to some pre-specified date pattern. Two examples of the pattern are
  ((\d{2})|(\d))[/.-]((\d{2})|(\d))[/.-]((\d{4})|(\d{2}))\s\-?\s** to match 12-08-2013 and
  ((\d{1,2})([/.-]?)(\d{1,2})?)?\s(st|nd|rd|th)?\s*(Jan(uary)?|Feb(ruary)?|Mar(ch)?|Apr(il)?|May|Jun(e)?|Jul(y)?|Aug(ust)?|Sep(tember)?|Oct(ober)?|Nov(ember)?|Dec(ember)?)* to match 12th Aug.
- Tag validation: Criterion fails if a date is found in a header tag, i.e. H*n* where *n* is an integer. A date commonly presented as normal text.
- Number of words: Criterion is met if the extracted string is not more than 20. If it is a free text (more than 20 words), the keyword must be part of the paragraph/division.
- Digit validation: This criterion ensures that the extracted data has at

least a digit (e.g. 2 June) and if it is more than 2 digits, then to verify the day, the month or the year is/are valid. If this criterion is met, fitness score is incremented by 1.

- Relevance and low-relevance words lists: Similar to the previous definition, these two lists contain positive (e.g. the name of the month) and negative words (e.g. 12hrs) relating to the valid date.
- Expressed after keyword: Keyword such as Date, Commence and Start normally precedes the actual date.

Course Price

Price can be defined as real numbers (with or without 2 decimal digits), which may be preceded or ended with a currency symbol or abbreviation. Another criterion is the valid length of the price. This should be able to isolate the price from other irrelevant numbers such as telephone number and course number.

The regular expression fitness algorithm is expressed as follows:
fitnessScore is equal to the number of successful criteria from the following
i) valid regular expression pattern, ii) text pattern, iii) tags, iv) number of words, v) digit validation, vi) word lists, vii) after keyword.

If the following criteria are satisfied, each contributes a point towards the total fitness score. Below is the detail of the criteria:

- Valid regular expression pattern: The regular expression is evaluated and if it is a valid expression, a score of 1 is added to the fitness score.
- Text pattern: The fitness is incremented on a condition that the extracted data must matched the valid text pattern such as *£\d+* and *s*((£)|£)\s*( )?\d+(,\d{3})*(.\d{2})?\s**
- Tag validation: The valid course price can be presented anywhere within the <body> tag except in a header tag.
- Number of words: Fitness score is incremented by 1 if the length is not more than 3 words or if it not more than 50 but having valid pattern.
- Digit validation: the extracted data contains between 2 to 4 digits

- Relevance and low-relevance words lists: Some of the relevant words are *vat*, £ or *GBP* and the examples of low-relevance are postage, brochure and materials.
- Expressed after keyword: valid keywords are price, fee or cost.

A 'Clean Grammar' approach

Automatically building regular expression patterns to extract title, location, date and price from training course web page aims to solve the scaling problem faced by hand coding regular expressions. Figure 5.2 shows a section from the 'clean grammar' in BNF and Figure 5.3 is its equivalent in XML format (see Appendix 3 for the full representation in this format). The same DTD definition used in the CoPE is also applied here to ensure the XML rules follow the specified syntax. In this grammar, the non-terminals refer to the HTML tags while terminals are the text string.

The basic grammar of regular expression is a sequence of patterns to be matched, where the patterns are expressed using literals (to be matched exactly) and pattern operators like '.' , '*' , '[' , ']' , '(' , ')' etc. However, in this work, rather than force the GP to build all the patterns from first principles, common idioms are used such as ".*?" as units.

The regular expression patterns have the general structure of:

Prefix (Content) Postfix.

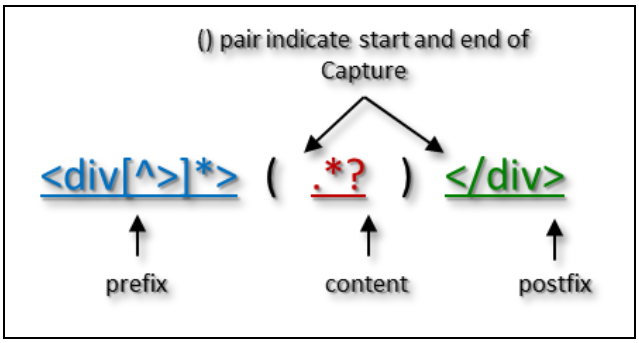


Figure 5.2. General structure of regular expression pattern.

Start	*::=*	*REpattern*
REpattern	*::=*	*opentag innercontent closetag*
opentag	*::=*	*"<" tagname "[^>]*>"*
closetag	*::=*	*"</" "N1" ">"*
tagname	*::=*	*"div" | "table" | "tr" | "td"*
innercontent	*::=*	*datacontent | REpattern*
datacontent	*::=*	*capturedata | tag_and_value*
tag_and_value	*::=*	*opentag innertags datacontent innertags closetag*
innertags	*::=*	*empty | singletag | singletagseq*
singletag	*::=*	*opentag closetag*
singletagseq	*::=*	*singletag innertags*
capturedata	*::=*	*"\s*?" "(" expression ")"*
expression	*::=*	*".*?"*

Figure 5.3. Regular Expression Grammar rules expressed in BNF form. In the actual implementation, these grammar rules are presented in XML format.

This means the pattern is trying to find the content in the context of prefix and postfix. For example, Figure 2.24 shows the structure for the pattern generated by GP; <div[^>]*>(.*?) . Therefore, if this pattern is applied to the HTML page below, "This text is extracted" found in between the first pair of "div" will be extracted.

<html>
<body>
* This text is extracted *
* This is another text *
</body>
</html>.

Evaluation Performance Strategy

Evaluation is based on how good the GP system is in relation to the number of required generations to produce a regular expression for successful extraction.

In the context of this part of the research (this chapter), only precision rate is calculated as the extraction is calculated based on the number of generations required to find the successful regular expression.

5.2.4 Regular Expression based on Extraction Rules

In normal circumstances, the key representative features of particular information can be expressed in a standard construct of regular expression. Some basic regular expression notations representing the DOM tree such as TABLE, TR, TD, DIV etc. as in figure 5.4 below are hand coded in the XML file as an initial pattern to guide the GP method. This should provide a baseline for the GP to start with. A new set of notation may be added later once it is found in the training set provided by the user.

For the purpose of this thesis, the regular expression grammar (Section 3.2) for the GP used in Xhemali's automatic WIE has to be reconstructed as it is hard to read or add new rules. This is because building regular expressions should start from the pre-defined component rather that the first that appears in the file and the elements of the rules should be referred by a unique name rather than numbers. By specifying the starting point, flexibility to read the file by another program can be achieved. Also the previous file structure does not provide the structure validation tool such as DTD to ensure that the rules conform to the correct syntax. The purpose of this XML file for this research is repeated here, i.e., to guide the generation of regular expression pattern to match the pattern of the relevant information for extraction.

```
<grammar>
 <start>
        <nonterminal name=”REpattern” />
 </start>
 <rules>
        <rule name=’Repattern’ type=’sequence’>
                <nonterminal name=’opentag’ />
                <nonterminal name=’innercontent’ />
                <nonterminal name=’closetag’ />
        </rule>
        <rule name='tagname' type='selection'>
                <token>h1</token>
                <token>h2</token>
                 <token>div</token>
                <token>table</token>
                <token>tr</token>
                <token>td</token>
                ….
        </rule>
        ….
        <rule name='capturedata' type='sequence'>
                <token>\s*?</token>
                <token>(</token>
                <nonterminal name=”expression” />
                <token>)</token>
        </rule>
        ….
</grammar>
```

Figure 5.4. An extract of *Grammar definitions* to evolve regular expressions.

The size of a regular expression varies. It can be as simple as a line long or it can be a full size of A4 paper if it needs to perform a complex matching. Xhemali (2010a) argues that the chromosome size of the regular expression should vary as it is far from structured. However, in this research, using a fixed-block size genome still applies. The block size for each rule is determined by the rule which requires the most non-terminals. Unlike CoPE, the resulting extraction pattern could vary in length. This is because when a particular block in the genome mapped the ‘capturedata’ rule (Figure 5.5), the mapping process terminates and the remaining unused blocks are ignored. The shortest pattern that can be produced from the mapping is a single tag, for example, <div[^>]*>(.*) .

Following the success in the CoPE experiment, the repairing function to ensure the syntactic correctness of the evolved regular expression is also not required here. This is because the grammar is built in such a way that the

pairing of tags is properly defined in the grammar.

Experiment and Results Discussion

In order to test the Regular Expression Evolution (REGEXEV) system, a suitable set of web pages must be selected. The web pages chosen should be able to concisely represent key aspects of very diverse structured web pages containing diverse data format. To evaluate the idea, REGEXEV system was tested on a number of web pages from training course websites, listed in Table 5.6. Only the first nine URLs are taken from Xhemali (2010a), while the remaining URLs are randomly selected. This is because some of the URLs are no longer available or accessible and in some web pages the required data types (attributes) are less than three.

Table 5.6. Sample websites to test the algorithm.

#	URL	#	URL
1	www.underoak.co.uk	**9**	www.chesterfield.ac.uk
2	www.ptp.co.uk	**10**	www.itleaders.co.uk
3	www.managementtrainingcoursesuk.co.uk	**11**	www.adepttraining.org
4	www.trainanddevelop.co.uk	**12**	www.findcourses.co.uk
5	www.dncc.co.uk	**13**	www.qa.co.uk
6	www.reedtraining.co.uk	**14**	www.campdenbri.co.uk
7	www.ontargetlearning.co.uk	**15**	www.beauty-school.co.uk
8	www.challengeconsulting.co.uk	**16**	www.medicalinterviewsuk.co.uk

There are a total of 48 sample web pages tested from all URLs to ensure the consistency of extraction result. For each URL, ten random seeds are applied to calculate the average generations required to reach a successful solution. The sixteen training course websites are chosen for the test varying significantly with respect to (i) the record size on the page, (ii) HTML tags used to hold the attribute content, (iii) data format and (iv) the content complexity, i.e. multiple instances attributes (such as a web page containing prices showing a standard course price and a discounted price).

In the experiment, REGEXEV operates as follows:

1. Execute the system at the command prompt specifying the seed value and the URL of the website.

2. An initial population is created and the genotype-phenotype mapping process is applied and the successfully evolved regular expression is applied to the web page to determine its fitness.

3. The fitness evaluation is examined using the specific set of criteria.

4. Ten individuals of higher fitness score are carried forward to the next generation for the reproduction process.

5. The system terminates if the solution is found or the maximum number of generations is reached.

Initially, the fitness of the generated regular expression was based on two criteria:

a. An algorithm to ensure that only valid regular expression is generated.

b. The output matches the pre-set format of particular information.

This simple analysis was chosen for processing efficiency that avoids a comprehensive knowledge intensive analysis. Unfortunately, the first observation indicates that the system also picks up inaccurate information. Because the useful course details have dissimilar format, generalising the data pattern is not possible. So five more criteria (c to g below) are added that would allow the system to recognise the attributes more effectively, thus improving the result:

c. Valid location: Location is made up of a number of strings. Generally a location is much harder to identify compared to other fields if match is dependent on the format. However, in this research, a location is considered valid if it is located in the UK, which names are stored in a file. At the moment this list of places in the UK is manually coded. For the future, this should be validated using a gazetteer.

d. Match title field: Because a title is made up of text, thus test on the general format is not helpful. A title match is considered true if it equals to the title field, i.e., text enclosed between <title></title> tags. Although, it is not always possible to take title field as the title due to issues mentioned in Xue et al. (2007), however, in this thesis, if a text within the HTML body is found 80% similar to the title field content, then it will be accepted as true. MinHash algorithm using Jaccard's similarity coefficient (Jaccard 1902, 1912), defined as $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$, is used to estimate the degree of similarity. Here, the number of matched strings in both sets divided by the total number of elements in both sets makes an index. The index equates 0 when both sets are dissimilar and 1 for exact match. In all other cases, the index is scored strictly between 0 and 1. This means the two strings are more similar if the index is closer to 1. True match normally is sensitive to the case of the letters, thus both strings are converted to lowercase before the comparison takes place. Because the title involves a longer string, Jaccard similarity coefficient is more appropriate as it is a token-based measure as opposed to character-based such as Jaro-Winkler distance (Winkler 1990) and Levenshtein distance (Levenshtein 1966).

e. Valid length: it is assumed that a title or a location of a course is valid if it contains not more than twenty words and if there are more (in case the course attribute resides in a paragraph), the attribute must be preceded with the keyword associated with that attribute.

f. Character length: Based on the analysis of the web pages from the 45 websites, it is concluded that this criterion is satisfied if the number of characters in each extracted word is no longer than 20.

g. Relevance and low-relevance corpuses: A set of relevant and low relevant words are devised to provide an indicator to the system for true positive or false positive data. The relevant words contains the set of keywords created from the context string and low-relevant

words are irrelevant words, such as, our, delivery and contact. The inclusion of these two categories appears to be very useful to reject the extraction of the incorrect attribute.

A regular expression pattern is evolved by applying the Genetic Programming method described in Section 4.3.1 using the standard genetic operators (selection, crossover and mutation) for a fixed number of generations until it converges or reaches an optimal solution. At this stage, the extraction rules the experiment uses are fixed and require manual updating. This is not an ideal solution, as updating new rules specifically requires a regular expressions expert.

Although the existing rules might be successful on a large number of web pages in the domain, they would not be able to cope with all other possible regular expression combinations (presumably containing other notation than those in the file) that can be successfully applied to other newly discovered/changed web pages. However, later through the TS-WIE system, this will be incremented automatically, with the HTML tag names and data format are more likely to be affected by the update. Incrementing rules with newly discovered patterns improve the chances of producing new extraction patterns thus increasing the success of the extraction: recall and precision rate. Table 5.7 shows some examples of the evolved regular expressions that matched the relevant data of a course. These regular expressions were applied to the web pages from the training course website listed in Table 5.6.

Table 5.7. Successfully evolved regular expressions to match the data of a course.

#	attributes	Evolved Regular expression
1	title	<title[^>]*>\s*(\b([A-Z]\s)?([A-Z][a-z0-9]+))</title>
2	title	<h1[^>]*>\s*([A-Za-z0-9]+\s+[A-Za-z0-9]*)</h1>
3	price	<div[^>]*>\s*((£)|£)\s*( )?\d+(,\d{3})*(.\d{2})?\s*(GBP)?)
4	price	<td[^>]*>\s* ((((£)|£)\s*)?( )?\d+(,\d{3})*(.\d{2})?\s*(GBP)?(\+\s*VAT)?)</td>
5	location	<li[^>]*>\s*(\b([A-Z]\s)?([A-Z][a-z0-9]+))< /li>
6	date	<td[^>]*>\s((\d{1,2})([\/.-]?)(\d{1,2})?)?\s*(st|nd|rd|th)?\s* (Jan(uary)?|Feb(ruary)?|Mar(ch)?|Apr(il)?|May|Jun(e)?|Jul(y)?|Aug(ust)?| Sep(tember)?|Oct(ober)?|Nov(ember)?|Dec(ember)?\s+(\d{2,4})?)</td>
7	date	<td[^>]*> \s*(\d{1,2})[/.-](\d{1,2})[/.-]((\d{4})|(\d{2}))</td>

Unlike Xhemali (2010a), the technique used here is using a pre-set data format, which acts as the initial pattern to match the relevant data, for example, a regular expression to match the price is "((£)|£)\s*( )?\d+(,\d{3})*(.\d{2})?\s*(GBP)?". The idea is to reduce the unnecessary evolution time evolving a regular expression for a strongly structured object.

Here, the evaluation is done by comparing manual extraction result by human against the output of the REGEXEV system. It is assumed that extraction by a domain expert is 100% accurate. The effect of the translation from genome to phenome in relation to the extraction rules and the fitness test is studied. For each URL, the GP was seeded with ten random numbers. The result of this initial work is reported in Table 5.8. Column 3 shows the percentage of precision for the 10 seeds. For example, in (b) for url #2, on average, only 50% of the seeds managed to extract the correct instance of the date. Column 4, 5 and 6 show the best, average and median precision of the successful seeds in the form of the generations the system requires to reach the first hit achieved from the 10 random seeds to find the optimum solution.

Table 5.8 (a-d). Result of matching regular expression to the course attributes based on the % of hits, best generations, average generations and median generation applied to each URL. Note that '-' indicates that the attribute is not available on the web page. The 'managementtrainingcoursesuk.co.uk' attributes (excluding title) comes from a linked webpage; external to the current webpage of interest. 'ontargetlearning.co.uk' and 'medicalinterviewsuk.co.uk' locations are labelled using some specific names of particular places other than the city name e.g. hotel name and room name, which failed to be recognised by the system.

(a) Title extraction

#	URL	% Hits (seeds)	Generations (Title evolution)		
			Best	Avg	Med
1	www.underoak.co.uk	100	0	32	28
2	www.ptp.co.uk	100	1	63.1	17
3	www.managementtrainingcoursesuk.co.uk	100	0	13.2	13.5
4	www.trainanddevelop.co.uk	100	0	8.6	1
5	www.dncc.co.uk	100	0	0.2	0
6	www.reedtraining.co.uk	100	0	4.6	2
7	www.ontargetlearning.co.uk	100	0	8.8	0
8	www.challengeconsulting.co.uk	100	0	4.7	0.5
9	www.chesterfield.ac.uk	100	0	0	0
10	www.itleaders.co.uk	87	0	18.7	1
11	www.adepttraining.org	100	0	8	0
12	www.findcourses.co.uk	100	0	1	0
13	www.qa.co.uk	70	0	0.4	0
14	www.campdenbri.co.uk	80	0	0.8	0
15	www.beauty-school.co.uk	100	0	0	0
16	www.medicalinterviewsuk.co.uk	100	0	0	0

(b) Date extraction

#	URL	% Hits (seeds)	Generations (Date evolution)		
			Best	Avg	Med
1	www.underoak.co.uk	100	0	25	21
2	www.ptp.co.uk	50	0	2	3
3	www.managementtrainingcoursesuk.co.uk	0			
4	www.trainanddevelop.co.uk	100	0	6.1	3
5	www.dncc.co.uk	100	0	1.3	0
6	www.reedtraining.co.uk	100	0	4.4	5
7	www.ontargetlearning.co.uk	-	-	-	-
8	www.challengeconsulting.co.uk	70	0	2.8	1
9	www.chesterfield.ac.uk	100	0	7.4	6
10	www.itleaders.co.uk	-	-	-	-
11	www.adepttraining.org	100	0	8	1
12	www.findcourses.co.uk	-	-	-	-
13	www.qa.co.uk	53	0	5.3	1
14	www.campdenbri.co.uk	73	0	38.6	7
15	www.beauty-school.co.uk	100	0	12.3	3
16	www.medicalinterviewsuk.co.uk	100	0	6.1	6

(c) Location extraction

#	URL	% Hits (seeds)	Generations (Location evolution)		
			Best	Avg	Med
1	www.underoak.co.uk	100	0	14	8
2	www.ptp.co.uk	100	0	3.5	0
3	www.managementtrainingcoursesuk.co.uk	0			
4	www.trainanddevelop.co.uk	100	0	8	8
5	www.dncc.co.uk	100	0	7.6	4
6	www.reedtraining.co.uk	-	-	-	-
7	www.ontargetlearning.co.uk	7	110	110	110
8	www.challengeconsulting.co.uk	20	0	0.33	0
9	www.chesterfield.ac.uk	100	0	23.9	7.5
10	www.itleaders.co.uk	-	-	-	-
11	www.adepttraining.org	100	0	5.6	4
12	www.findcourses.co.uk	93	0	30	13
13	www.qa.co.uk	97	0	8.31	3
14	www.campdenbri.co.uk	100	0	14.9	3
15	www.beauty-school.co.uk	-	-	-	-
16	www.medicalinterviewsuk.co.uk	0			

(d) Price extraction

#	URL	% Hits (seeds)	Generations (Price evolution)		
			Best	Avg	Med
1	www.underoak.co.uk	100	1	16	13
2	www.ptp.co.uk	63	0	3.8	0
3	www.managementtrainingcoursesuk.co.uk	0			
4	www.trainanddevelop.co.uk	100	0	13.9	2
5	www.dncc.co.uk	100	0	2.7	0
6	www.reedtraining.co.uk	100	0	3.9	0
7	www.ontargetlearning.co.uk	100	0	5.9	0
8	www.challengeconsulting.co.uk	100	0	0.7	0
9	www.chesterfield.ac.uk	100	0	24.2	8
10	www.itleaders.co.uk	73	0	8.9	6
11	www.adepttraining.org	100	0	4.3	0
12	www.findcourses.co.uk	77	0	5.4	0
13	www.qa.co.uk	100	0	3	0
14	www.campdenbri.co.uk	100	0	18	10
15	www.beauty-school.co.uk	100	0	12.7	4
16	www.medicalinterviewsuk.co.uk	100	0	2.1	0

Out of the sixteen websites, 81%, 61%, 54% and 75% achieved 100% precision for the title, date, location and price respectively. The algorithm performed better on the *title* field compared to other fields. This means that the algorithm is more precise on the title. It has been observed that most of the titles are presented in header <h_n> tag. As can be seen, the experiments hit the correct attributes immediately, with some needing the rerunning of the

GP to achieve this best precision. Despite its success on identifying single occurrences of the attribute, the extraction algorithm performance on a web page containing a multi-value field illustrates its limitation. Also, it has been observed that the algorithm is poor when handling nesting structures and duplication of values. The following explains further on the limitations based on the results collected from the experiment (Table 5.8):

url#2 – Extracting many occurrences of the same entity in a web page is typically difficult. In a course offered, the price, which is unique for a given course, is listed several times depending on the location. Also, the course provider displayed a promotion of their different courses, which includes prices. This is confusing for the system to decide the correct price to extract.

url#3 – The precision is 0 for the date, location and price of the course offered. This is because these details are external to the HTML document and imported using an 'iframe'. If the web page is originating from a different domain, access to manipulate its iframe content is not permitted to a browser-side programming language (such as JavaScript) because of security reasons (W3C, 2010).

url#7 – Out of the 10 seeds, only seed 2 produced a successful solution for the location. The analysis of the web page shows that the data are located within a large text. Although this text contains the relevant attribute, it is counted by the system as an error because it exceeds the allowed text length.

url#8 – The system captured incorrect information as there is more than 1 date (date includes the information about course flyers). The system only achieved 20% precision for the location as the web page not only displays the venue of the course, but it also displays the organisation's office address, which is also a 'valid' location.

url#13 – The algorithm performs poorly when there is a duplicate of attribute instances e.g. the web page 'last updated' field, which normally appears in the footer section, is misinterpreted by the system as the date of the course because the format is valid for a date. This could

be avoided if the extraction is confined to the main content only as proposed by researchers of content extraction.

url#16 – The system did not successfully recognise the correct location on this web page. There are a few cities mentioned on the page but the correct location is the name of a hotel or a specific name of the training room, thus, difficult for the system to decide on its relevance to the extraction.

It has been observed that firstly, the GP system does not need to use all the blocks to reach a fit solution. For example, the title is found only using one block of the 10 blocks genome since most of the titles are presented in header tags (e.g. h1).

Secondly, irrelevant contents such as navigation and advertisements were identified within some of the web pages within the same website. The same performance was observed from the successfully generated regular expressions for these web pages. This implies that these irrelevant contents do not have any effect on the accuracy of the web page extraction.

Thirdly, the proposed technique is able to correctly extract key content from the majority of the web pages within the same website and the accuracy of the result is consistent. It is common for a website, which is designed by a developer or a team of developers, to use similar structure and data format to present the same data content, in this case the course training data. For example, all web pages in www.underoak.co.uk present the list of interrelated courses in a table.

Fourthly, the proposed technique also captures incorrect results. This is a result of a number of unknown words, which do not appear in the relevance corpus.

Fifthly, it is almost unfeasible for the algorithm to identify the relevant attributes from some long sentences, which are indicated by some kind of keywords e.g. ‘Date: ’. And finally, in some cases, the proposed technique

failed to capture any relevance data within the allowed set time. It has been observed that these data are not known due to the fact that they do not appear in either the relevance or the less-relevance corpus. Also, this data comes from a linked page presented in 'iframe' within the HTML document and the absence of HTML tags in the grammar.

The algorithm works well on properly structured and properly formatted web pages such as URL #1 and #11. An enhancement to the extraction method has improved the results and this is discussed in Chapter 6.

5.3 Related Work

The aim of this section is not to provide an in depth survey of the state of the art approaches. For such a survey, refer to Muslea (1999) and Ferrara et al. (2013). Since the problem of regular expression evolution in WIE is quite distinctive from the issues mentioned in the field of IE, the related work is discussed more detail here instead of in Chapter 2 which is dedicated to various IE approaches in general.

An approach that uses regular expression to extract information is proposed in (Li et al. 2008). Although this approach may be applied to a wide range of entity extraction tasks, the generation of the regular expression is not based on an evolutionary approach. It emphasises some sets of examples and some knowledge of regular expression, therefore demands a skilled user. Another approach to generate regular expressions to extract text information from the Web is described in Bartoli et al. (2012).

This section discusses the REGEXEV performance against a system proposed by Bartoli et al. (2012). Similar to REGEXEV, the research focuses on evolution of regular expression using GP for WIE. They aimed to extract HTML headings (Text appears in HTML tags <h1> to <h6>) and phone numbers. However, the extraction of phone numbers is beyond the purpose of

this research. The individual is only tested based on the extraction of the web page's header, referred to as title in this research.

This research differs from Bartoli's work in several ways. First, one major difference is that their research requires user intervention to provide a few tens of examples at the beginning of the extraction process. This allows their system to learn from these examples to automatically generate the regular expression. The regular expressions are encoded directly as program trees. Of course, the method requires a large amount of labelled examples to improve precision and recall. In contrast, this research uses the 'clean grammar', instead of asking the user to label the information by hand for the same purpose. Second, the training example set is composed of positive and negative examples. Each example set is made up of two strings; one text line and one substring from the line that must be matched by the regular expression. An empty substring indicates a negative example. The approach used in this research, on the other hand, does not require any training example. It is purely based on combination of individual HTML tags and textual pattern matching. This is because the tags are normally used to express the different elements in the web page, such as <title> indicates the title of the web page. Third, the fitness score is calculated based on the sum of the Levenshtein's (1966) edit distance between each detected string and the corresponding examples, and the length of the regular expression. This edit distance is used to find similarity between relations. On the contrary, this research uses Jaccard's Similarity Coefficient to find the similarity as it uses less calculation to produce the required result. Finally, the grammar uses character by character evolution. This would increase the search space and demands for high evolution time. The 'clean grammar' used in this research reduced this unnecessary consumption by handling the dataset as a token rather than a character set, where any word contained within the opening and closing tags is translated to .*? in regular expression.

Bartoli et al. evaluate their system using the datasets obtained from the publicly available web pages; extraction of information from Wikipedia and W3C websites. Their experiment used 49513 lines of HTML source from

these websites and these lines were split into i) 505 positive and 48608 negative testing sets, ii) 151 positive and 149 negative lines as training examples, and iii) 50 positive and 50 negative examples as validation sets. The regular expression successfully evolved was $<h\backslash d[\hat{}Z]++$, resulting in 96.1% precision.

Using the same websites, the result of the experimental study in this research to extract the web page title indicates that, for typical corpora like W3C and Wikipedia, a strictly hierarchical approach would indeed work. This is because of the similar template applied to all their respective web pages. In addition, the extracted text is further verified using the similarity test with the content of the HTML <title> tag. The regular expression successfully evolved in this experiment has a pattern of *<hn[^>]*>(.*)</hn>* where *n* is a digit. This demonstrates that a proper regular expression is produced that strictly extracts the text between the opening and the closing header tags. However, this is not the case for Bartoli's regular expression <h\d[^Z]++ which reads as opening header tag followed by any number of characters that is not Z. Such an expression will capture anything after the opening header tag until the end of the line and this data might be incorrect data (if another attribute follows the header tag on the same line) or incomplete data (if longer title is involved or the title contains uppercase Z).

5.4 Chapter Summary

This chapter has outlined the development and evaluation of regular expression evolution system. The regular expression evolution aims to build a data pattern to match the relevant data in a HTML web page.

It is interesting to see that the 'clean grammar' concept (Chapter 4) can be applied to other domains than programming subsets avoiding the repair function to ensure that the syntax is followed. The experiment evolving regular expressions showed mixed results. The perfect precision rate was achieved if the web pages are structured and the attributes (title, date, location and price)

of the courses are appropriately formatted. The system fails if the attributes are in linked files and it also struggles to make a correct decision if there are more than one identical data available on the page. However, the involvement of a human to teach the system to identify the correct attributes indicates an improved result (see Chapter 6).

Chapter 6

Practical Application of Teachable Semi-Automatic WIE

6.1 Chapter Overview

The previous chapter described an extended GP method based on a 'clean' XML-based grammar definition for regular expression evolution to discover information on web pages. It has become apparent that the (automatic) GP technique using the combination of the DOM tree and the data format to identify instances on the web pages has its limitations especially when multi-instance attributes are involved. The limitations suggest that there will always be a need to involve a human domain expert to teach the Web Information Extraction (WIE) system.

This chapter is rather technical; however, it includes the important contributions for the data extraction. It serves to fulfil objectives 5, 6 and 7 stated in Chapter 1. The novel approach introduced is the implementation of DOM tree in jQuery rather than the common XPath. Particularly, in this chapter, the model and the methods for a teachable WIE (TS-WIE) are presented. It aims to solve the issues specified in Chapter 3 and improve the results in Section 5.2.4.

It is also to be noted that unlike other fully-fledged semi-automatic extraction solutions, the TS-WIE system is concerned with a single web page extraction rather than multiple web pages in a website. This decision is due to the fact that the purpose of this thesis is to provide a solution to improve the quality of the data extraction by an automatic WIE system employing GP principle that evolves regular expressions – as demonstrated in Xhemali's automatic WIE (*automatic WIE* for now). Furthermore, the relevant URLs in the database as a result of the Xhemali's crawler system (*crawler* for now) are the addresses of the individual relevant web pages that need further processing.

The next section introduces the target schema and Section 6.3 describes the methodology and tools used in the development of this TS-WIE system. Section 6.4 and Section 6.5 introduce the basic components of the TS-WIE model and the elaboration of the architecture and the interface of the system respectively. In addition to the basic processes inherited from Xhemali's automatic WIE architecture, the TS-WIE system provides a refined extraction rules structure and an option to add newly discovered data patterns into the XML file that extends the extraction rules. Section 6.6 describes the evaluation of the system and the challenges faced by the system are outlined in Section 6.7. Before concluding the chapter, the REGEXEV was revisited to assess the effect on its performance after the changes made to the grammar collection by the TS-WIE system.

6.2 Target Schema

The purpose of the *Crawler* (Xhemali 2010a; and also described briefly in Section 3.3.1 of this thesis) is to provide a facility, which selects only the relevant web pages from the training course domain that the TS-WIE system is to handle and hence reduces manual effort of searching and determining their significance. In the world of TS-WIE system, these selected web pages are presented to a user and the user acting as the expert defines the relevant information for the system to process. Table 6.1 shows the URLs, which have been used to test and evaluate the technique proposed for TS-WIE in this research. The first nine websites are taken from the list in Section 5.2.4. The remaining websites are chosen from the websites of training courses operating in the UK. The aim of the TS-WIE system is to discover and generate a set of new extraction rules based on the set of training examples provided by the user. These rules are useful to compare the new performance of the extractor described in Section 5.2.4 after this new discovery.

Table 6.1. List of URLs relevant to test and evaluate the TS-WIE System to improve the quality of the extracted information.

#	URL	# of relevant web pages	#	URL	# of relevant web pages
1	www.ptp.co.uk	372	**15**	www.capita-ld.co.uk	152
2	www.managementtrainingcourses uk.co.uk	5	**16**	www.cim.co.uk	129
3	www.trainanddevelop.co.uk	46	**17**	courses.independent.co. uk	5506
4	www.ontargetlearning.co.uk	37	**18**	www.coursesplus.co.uk	1929
5	www.challengeconsulting.co.uk	30	**19**	eca.co.uk	23
6	www.itleaders.co.uk	3	**20**	register.rit.edu	220
7	www.findcourses.co.uk	4680	**21**	www.cipd.co.uk	190
8	www.campdenbri.co.uk	127	**22**	pgplus.bisgroup.com	174
9	www.medicalinterviewsuk.co.uk	127	**23**	www.hemsleyfraser.co.uk	250
10	www.beauty-school.co.uk	82	**24**	www.ldl.co.uk	31
11	academyclass.com	61	**25**	www.locksmiths-training. co.uk	1
12	www.loucoll.ac.uk	278	**26**	www.theiet.org	102
13	www.skillsolve.co.uk	135	**27**	www.chesterfield.ac.uk	1369
14	www.spearhead-training.co.uk	237			

In case of the TS-WIE system, the system is expected to learn what it needs to know about the training data. It is only meant to handle the extraction of explicit information, without the need to understand the semantic of the extracted information. Furthermore, the training data does not need to be explicitly typed, which therefore reduces human input error.

6.3 Methodology Adopted

Generally, research methodology refers to the principles and procedures of logical thought processes which are applied to a scientific investigation (Klein & Myers 1999; Fellows & Liu 2003). In system development (where this study is concerned), a methodology refers to a collection of procedures, techniques, tools and documentation aids, which provide appropriate guidelines for the implementation of a new information system (Avison & Fitzgerald 2006).

The TS-WIE system is built following the methodology for Information System (IS) Research proposed by Burnstein (2000), which divides the IS development activities into three stages (Table 6.2); Concept Development, System Development and System Evaluation.

Table 6.2 The research methodology for the TS-WIE system.

PHASE	ACTIVITIES
Concept Development	Background Investigation • Explore research area • Investigate business processes • Investigate existing WIE system's functionality • Define user requirements Analysis • Analyse the user & business requirements • Analyse the domain area • Analyse existing WIE system's architecture, functions & data storage Design • Application design based on requirements • GUI design – web based application
System Development	• Investigate suitable programming language • Prototype TS-WIE development • Prototype GUI development • Test System – Unit & Integration

System Evaluation	• Evaluation of result; precision, recall and F-measure
	• Impact of the TS-WIE system to
	  ○ Training Course domain
	  ○ Automatic WIE system
	• Evaluate aim and objectives
	• Define contributions

The system development always relates to costs, performance, functions and reliability and the development models aimed to help reduce the problems associated with growing complexity of the software project such as escalating cost and late delivery (Van Vliet 2008; Granlien et al. 2009).

The TS-WIE system uses Software Development Life Cycle incorporating Prototyping (Figure 6.1) to ensure the system is developed within acceptable standards (Carey 1990) and help control the risk of incomplete requirements (Floyd 1984). It is considered suitable for this research for the following reasons:

i) The prototyping allows for an iterative process, which evolves to meet user requirements and at the same time increases the likelihood of user acceptance of the final system.

ii) It only concerns small application software involving low risk.

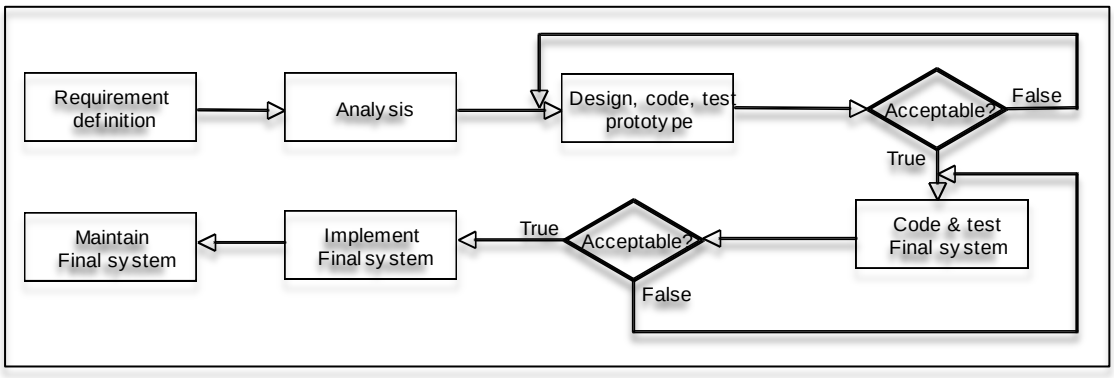


Figure 6.1. System Development Life Cycle with Prototyping (source Carey 1990).

Prototyping, as noted by researchers, enables the Information System development process to be broken down into small and manageable steps (Kraushaar & Shirland 1985). It also improves communication between the developers and the users (Alavi 1984) to cope with uncertainty (Granlien et al. 2009), cost effective (Boehm 1988; Palvia & Nosek 1990; Gordon & Bieman 1995) and encourages greater user involvement and participation in the development process (Naumann & Jenkins 1982). Carey (1990) states that the user interface is one of the three main areas that is often prototyped. This is also the case for this research to build an interactive web application because the user can clearly express the needs at the beginning and offer immediate feedback, which results in better interface design.

Requirements of the system

The following describes the important requirements of the TS-WIE system:

1. Extraction requirement.

   The domain is the UK course provider and the information which needs to be extracted are the course title, location, date and price. A comprehensive extractor needs to deal with web page issues such as missing and conflicting data, client-side JavaScript and other similar features.

2. Graphical User Interface (GUI).

   The system should provide a facility for a non-expert user to build a wrapper for the extraction through a user interface. The interface should be able to get some set of input data from the user. The input will be in the form of labelled data, which is used later to train the extractor system. The user should be allowed to provide input consisting of some or all of the required information, i.e., course title, location, date and fee (see Section 6.5.2 for detailed discussion on the design of user interface).

3. URLs from the database.

   The TS-WIE system should be able to validate the relevant URLs, which are stored in the database. After extraction process for a URL is successfully completed, its status in the database should be updated appropriately to avoid unnecessary repeat processing.

4. Uploaded web page.

   When a valid website is uploaded in the GUI for the user to make a selection, the web page must be as it would normally appear on the web browser. This should include elements such as buttons, images, formatted fonts and line breaks.

5. Integration with the existing automatic WIE.

   The TS-WIE system should integrate well with the existing automatic WIE system.

6. Update database records.

   Upon successful extraction of the specific information, several tables in the database should be updated, which are the CIE_Allowed_links, GP_Genome_Phenome, GP_Phenomes and GP_Genomes (refer to Figure 3.5, Section 3.3.1.2 for automatic WIE database design).

6.4 TS-WIE System Overview

This section introduces the TS-WIE model to support the extraction from unfamiliar web pages. Figure 6.2 shows the system view of the TS-WIE extending the existing automatic WIE. The TS-WIE (represented inside a grey box) is a semi-automatic system aimed at providing an interactive wrapper learning feature from unsuccessfully processed web pages, when the previously generated extraction rules fail to extract the relevant attributes.

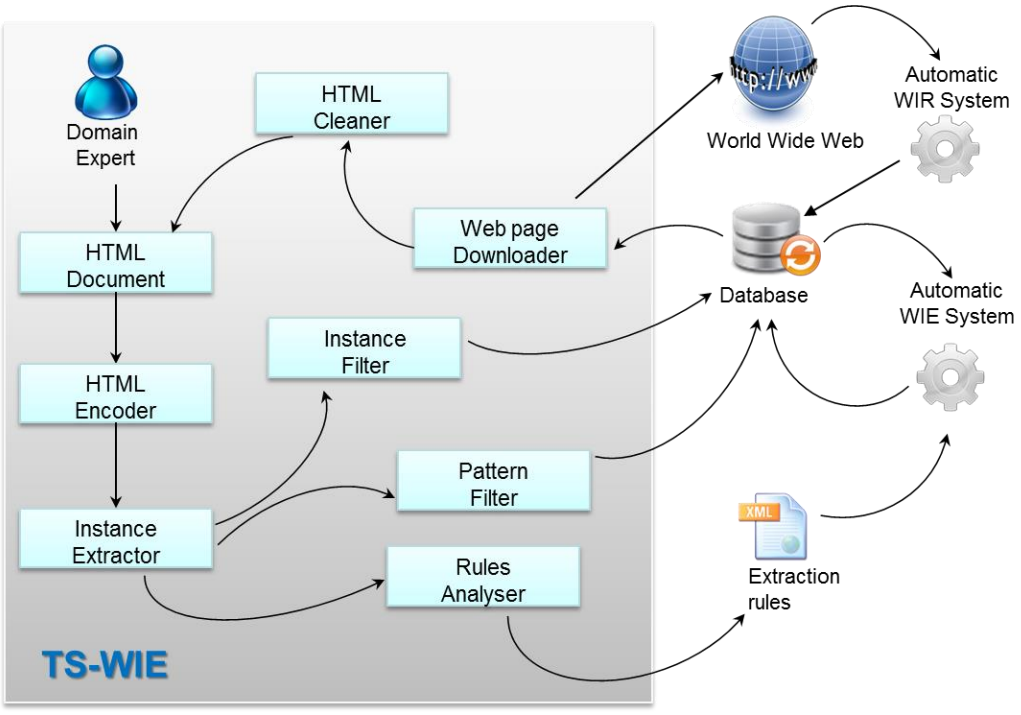


Figure 6.2. TS-WIE system supporting the automatic WIE system.

This TS-WIE system also supports the need to maintain the wrappers. This normally happens due to the previously processed web page evolving over

time, which cause the wrapper built for it to produce an inaccurate result. Other systems applying a similar approach have been discussed in Chapter 2.

In the view of functionality feature, TS-WIE contains three tasks; pre-processing task (web page downloader and HTML cleaner), Attribute Extraction task (HTML encoder and extraction task) and post-processing task (Instance Filter, Pattern Filter and Rules Analyser). The tasks are elaborated in detail in Section 6.5.3.

Pre-processing task: The *Web page Downloader* downloads the user's selected URL and if it is a valid URL, it is stored locally. A well-known cleaner tool called HTML tidy (Raggett 2012) is applied to deal with any structural errors. The next process is the *HTML encoder* to standardise the HTML attributes, such as dash (–, —) to their equivalent symbols and disables any hyperlinks to be able to work on that web page.

Attribute Extraction Task: After the user made the selection of example data, the extraction process begins. The *extractor* then grabs the selected example(s), i.e., the training data and the tag's traversal path(s) provided by the user. The user provides a single example to indicate single record extraction and two examples to indicate multi-record extractions.

Post-processing task: This task has three features; Instance Filter, Pattern Filter and Rules Analyser. The Instance Filter validates the instances for duplicates. Unlike the Instance Filter, the Pattern Filter is concerned with conversion of the instances and the paths into their equivalent patterns in the form of regular expressions and then validates these patterns. A special script was written to do this conversion. Rules Analyser takes the path, converts it into tokens and puts the new tokens into the XML file.

For practical reasons, PHP and JavaScript, using the jQuery library and JSON (for transmitting structured data – objects to the server application) were

chosen for developing the Graphical User Interface (GUI). The reasons are that these tools are suitable to handle client-based GUI application, and the researcher is familiar with them. A standard web server application (APACHE) was used in combination with PHP scripting as the server-side web programming, for communicating with the client and the data storage.

The data storage used to store the relevant course information is MS SQL Server 2000 and the extraction rules are in a XML file. Although, in general, MS SQL Server 2000 is not the best data storage available in comparison to PostgreSQL or Oracle, one of the requirements is to integrate with the automatic WIE at ATM, which uses this database, and the same database is also used by the Customer Relationship Management (CRM) software. Because an XML file was used to store the extraction rules in the automatic WIE and the core process of WIE (generation of regular expression) depends on these rules, this research does not attempt to evaluate the feasibility and efficiency of other storage systems as it will disrupt the running of the automatic WIE. Moreover, Xhemali (2010a) provided evidence that XML file has offered significant support to the extraction task.

The following sections discuss the TS-WIE system design and implementation, including the experiments and discussion of the results.

6.5 TS-WIE System Design and Implementation

6.5.1 Introduction

Researchers such as Cohen & McCallum (2003a) and Fernandez-Villamor et al. (2012) argue that using DOM tree alone in automatic extraction is insufficient to identify particular data on the web page. Cohen & McCallum states that combining DOM with the data pattern can only slightly improve the quality of extraction, thus he proposed the addition of a visualisation approach for further improvement. However, DOM manipulation is useful in the content extraction method, where the web page segmentation is used to identify the

main area for extraction (Cai et al. 2003; Kohlschütter & Nejdl 2008; Raavi & Somayajula 2012; Omer et al. 2012; Choochaiwattana 2012), which limits the search area and avoids the undesirable content.

The experiments in this thesis were conducted based on semi-automation to provide empirical evidence that DOM and data pattern performs as well as the combination of the above three techniques. Unlike Raeymaekers and Bruynooghe (2007), only two positive examples are needed to train the system. Negative labels and corrective mechanisms are not implemented as it is assumed that an example and the counterexample are sufficient for the system to produce an effective extraction pattern. Brin (1998) is the first to come up with an approach that reduces the training cost by just starting with a few 'seed' tuples to discover the extraction pattern, which could become new seeds for the next process iteration (called bootstrapping).

The dynamic generation of the wrappers (extraction patterns) based on DOM tree and data pattern using regular grammar combined with Artificial Intelligence has not been empirically proven in Web Information Extraction research. To the best of the researcher's knowledge, the common DOM tree solution implementations are using XPath and none applies jQuery notation. Here, jQuery is chosen due its simplicity and because it works well with javaScript; a popular scripting language for client-based web application. Other advantages of jQuery over XPath were discussed in Chapter 2.

The fundamental design is based on the following observations:

- Supervised methods are more accurate than unsupervised (Barbosa et al. 2013) and improve the quality of IE applications (Doan et al. 2008; Ferrara 2013). Therefore, the TS-WIE system is built to be interactive, thus a user interface to accept training example(s) from the user is provided.

- There is a trade-off between the number of training examples and the number of characteristics that the system can learn from the selected data. This means that requiring too many examples would burden the

user and requiring too few would give the system less matching quality power. Thus, this research attempts to balance this issue by only requiring two examples from the user to define the relevant data from the web page.

- It is assumed that the training examples provided by the user are accurate so that the system can generate the correct wrapper to extract other similar instances of a particular attribute from the same web page or from other similarly structured web pages within the same website or other websites.
- An extensible and flexible data model for representing the extraction rules is required to improve the quality of the extraction.

The design of the system is divided into two dimensions; a domain expert and a pattern expert. The *domain expert* is the human user who has knowledge of the domain and the extraction task. This user will teach the TS-WIE system where to locate some specific piece of information on the web page, whereas the *pattern expert* is algorithm, which is responsible for filtering the information, generating a useful extraction pattern based on the selected set of training data and adding any new extraction rules to the system's existing knowledge-base for future use. This section aims to answer the question of 'how effective is the extraction method to capture quality information and can it performed better than the automatic extractor system with the rigid grammar?'

In order to facilitate the input from a human (*domain expert*), the section below describes the user interface design. This is then followed by the discussion on the *pattern expert* design and challenges within the development of the TS-WIE system.

6.5.2 Graphical User Interface

The TS-WIE is a client-based system that provides a visual viewer in the form of a web browser-based user interface. The purpose of this editor is to allow a human to locate and identify a single example or two examples easily from the target web page. Two examples suggest that there are two or more occurrences of a particular attribute on the web page. The usage of this system does not require the user to be familiar with either the GP or the regular expressions.

According to Howcroft and Carroll (2000), many of the new methodologies were aimed at the look and feel of the user interface, which failed to address the wider aspect of web based information systems. The user interface can be designed in an optimal way using GUI elements such as command buttons, input boxes, drag-and-drop and highlighter. However, this thesis is not focused on creating the best interface for the semi-automatic system, rather it concentrates on providing the functionality required in an interface to accept input provided by the user and validates the response from the system that act on the input. Furthermore, the prototype model of the GUI (Figure 6.3 & 6.4) does not require a specific browser.

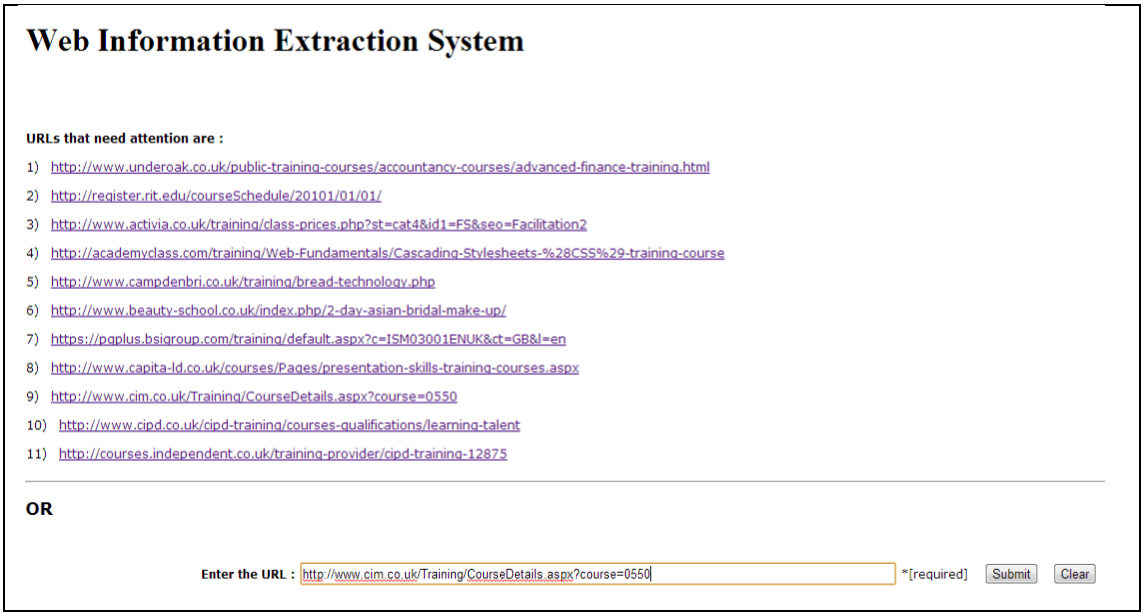


Figure 6.3. A screenshot of the user interface for selecting the URL to process.

The first page that the user will see is depicted by Figure 6.3. It contains a list of URLs that need attention. The user is given a choice of clicking the URL or typing the full URL in the box provided in order to work on that particular web page.

The list of URLs is dependent on the number of records in the database, which have been identified by the automatic WIE as unsuccessfully processed, thus need to be dealt with. The user is required to choose the URL from the list, for example, *http://www.capita-ld.co.uk/courses/Pages/absence-management-training-courses.aspx* and when the user clicks the submit button, the screen as in Figure 6.5 will appear. The actual web page is in Figure 6.4.

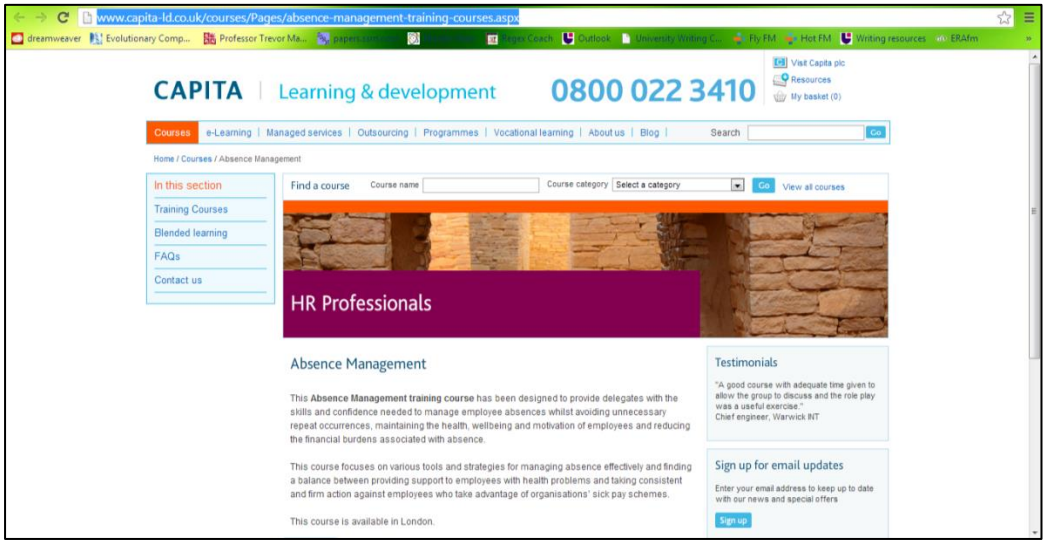


Figure 6.4. A screenshot of http://www.capita-ld.co.uk/courses/Pages/absence-management-training-courses.aspx web page rendered by Google Chrome.

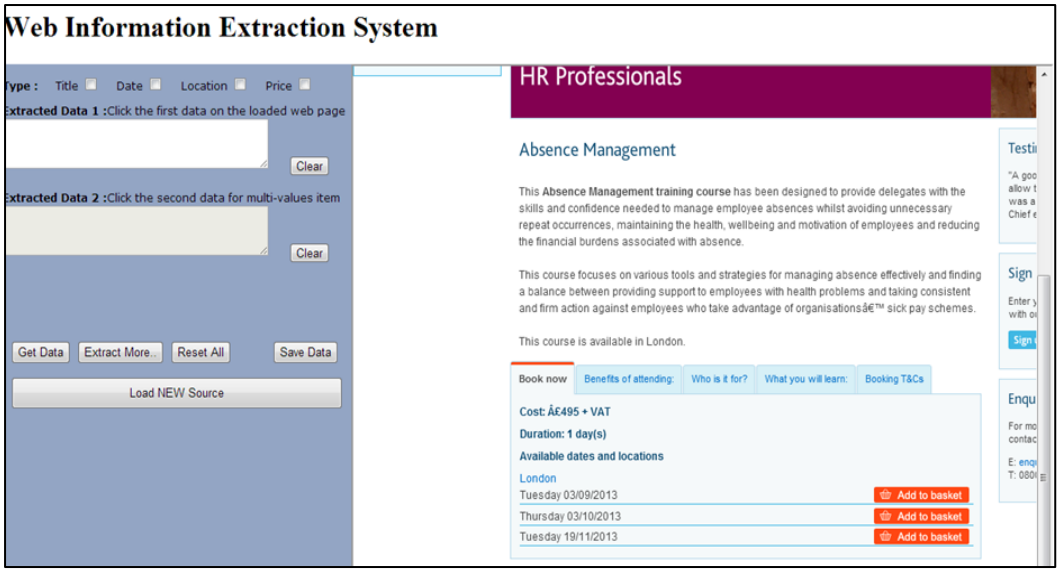


Figure 6.5. TS-WIE interface to accept example(s) from the user.

The user interface (Figure 6.5) with the function to accept examples from the user is divided into two main sections. The largest section (right side) is an area for rendering the relevant web page, which is scrollable. It allows interactivity, where the user can point and highlight the relevant information. The left section is the processing area, where the selected data are received and displayed. There are several process buttons in this area, which are described below:

- The **Get Data** button captures the single or multiple instances of the selected attribute and the corresponding attribute's path.
- The **Extract More..** button refreshes the left hand section to allow for the selection of the next attribute to be extracted.
- The **Reset All** button clears the input boxes.
- The **Save Data** button saves all the extracted data in JSON format to be processed further before permanently saving it in the database and XML file.
- The **Load NEW Source** allows for selecting another source listed in the previous web page (Figure 6.3) for a new extraction task.

The approach is first to parse the loaded HTML page into a Document Object Model (DOM) tree representation and assess this page based on this tree information. DOM provides the ability of manipulating the DOM nodes on the web page. As the user moves the mouse around the web page, the current element hovered over is highlighted. When the user clicks on the element, the capture process is started. JavaScript, using the jQuery library, retrieves the specific nodes and elements of the DOM tree forming the path from the root to the selected element to provide a set of extracted key content to the next process.

The characteristics of the sample element should provide enough evidence for the TS-WIE system to understand where the information is on the web page, the keywords associated with it and the pattern that it is made of. The choice of examples depends on human judgment to decide which information should be extracted. To ensure the correctness of the selection, the selected data are displayed on the left of the window. Intentionally, the path is hidden from the user view as this does not provide any useful information to the user. Finally, the selected data is assessed for validity before it is extracted and exported to the database for future query.

6.5.3 System Design

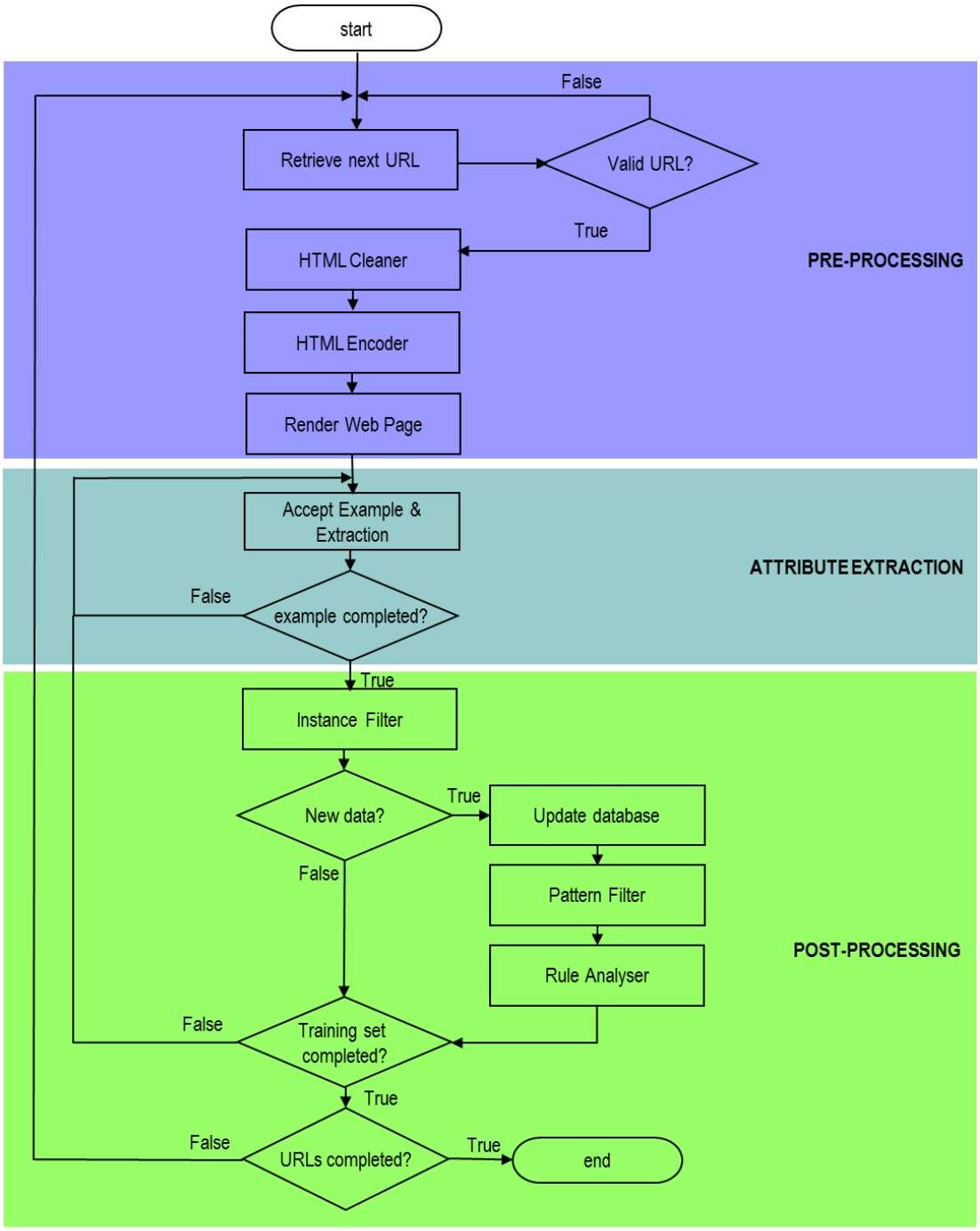


Figure 6.6. TS-WIE system in relation to flow of control between processes.

The TS-WIE consists of three main stages as depicted by the system flow in Figure 6.6:

1. Pre-Processing Stage.

   The URL, which has been previously retrieved by the crawler system and processed but unsuccessfully by the automatic WIE system, is retrieved from the database. The relevant data has not been extracted because the automatic WIE failed to recognise it. Each URL's status is indicated by its Link_status field, marked earlier by the automatic WIE (refer to Table 6.3 for the definition of each status integer). The URL is useful to the TS-WIE system if its Link_status is 5.

   The URL is first validated to ensure that it is still alive (if it is not, user will be notified) and then copied locally to avoid the 'permission denied' security issue (same origin policy[3]) due to different domain processing. This issue restricts the interfering with web pages belonging to other websites. The copied HTML document is set to be by default UTF8 encoded Unicode to avoid unnecessary warning with regards to I/O (like print).

Table 6.3 Link_status value, which is used to indicate the status of the relevant web pages retrieved. (Source Xhemali 2010a)

Link_status	Definition
0	The website is not used for training evaluation or classification
1	The website will be used for training
2	The website is used for training
3	The website has been evaluated and classified
5	The website is not successful

   The next step is to check and clean errors in the HTML document using HTML Tidy. The result of HTML tidy is an XHTML version of the web page. It is well known that one of the drawbacks of regular expression is that it cannot check for balanced tags and it will fail to match if there is inconsistency in the document. Therefore, in order to solve this, the

[3] See https://code.google.com/p/browsersec/wiki/Part2#Same-origin_policy

selected web pages need to be cleaned (structure fixing) first before further processing otherwise it may result in imperfect data extraction.

The HTML encoding is a special task which has 3 functions; specify a base URL, standardise the HTML attributes and disable hyperlinks.

Specify a base URL: The <base href> tag is embedded inside the <head> element to resolve several technical problems, such as handling different types of HTTP and HTTPS requests (especially those hyperlinked using relative URLs), such as images, content caching and AJAX execution. The XHTML web page is then uploaded into a frame ('iframe') in the system interface, which is necessary to preserve its presentation style formatting separate from the TS-WIE interface formatting.

Standardise the HTML attributes: It is possible that web designers use different HTML attributes to produce the same effect. For example (£) and £ will display UK currency symbol on most browsers. Therefore, to avoid creating various patterns to refer to the same thing, normalise representation is used. Other examples of HTML attributes are the non-breaking space ( ) and dash (– and —).

Disable hyperlinks: hyperlinks are links used to connect the current web page to another web page. If a hyperlink is clicked, normally it replaces the displayed web page with the target web page on the same window, unless the user instructs it to open in a new window. In training course domain, hyperlinks are commonly applied to the title of the course and the location, which navigate user to the detailed page of the course and to the specific location on the map. This movement of web pages make it impossible to grab the hypertext. Therefore, disabling all the hyperlink attributes ensures no interruption on getting detail of the selected information.

2. Attribute Extraction Stage.

The course attributes (title, date, price and location) to be extracted may only appear once (single instance) on the web page or there may be similar data (multi-instance). In a single record webpage environment, only one training data is required. However, where there are multiple records, two types of input are required; training data and validation data. The training data aims to provide a data pattern to the extractor system and the validation data is to confirm the pattern.

If two attributes are selected, the system assumes that other similar data items exist in the webpage and it expects that both selected data items must have a common DOM node (parent) and their patterns are the same if not similar, otherwise, the user is requested to reselect the second data item. The path collected is in the form of jQuery notation, which precisely identifies the position of the selected data item e.g. “html>body>div:eq(1)>table>tr:eq(1)>td:eq(1)”. Further detail is provided in the ‘jQuery Path Patterns’ section below and some basic information is in Chapter 2.

Multiple inputs will only be considered valid by the system if the training data formats are > 90% literally similar and the paths (parents) are literally identical. Jaccard’s (1902, 1912) similarity coefficient is used to estimate the degree of similarity between the two sets – first and second tokenised data pattern, and first and second parsed paths.

The data selected might be part of a long sentence. An algorithm was written to do a ‘Data Filter’ to refine this selection before the regular expression is generated. The long sentence is passed through a process called tokenisation. Tokenisation decomposes a sentence into tokens along a predefined set of delimiters (like spaces, commas, and dots). Then the relevant data pattern or relevant keyword is matched against the tokens to identify the important data. Note that this kind of presentation normally relates to the web page with single instance of attribute.

However, the same algorithm is also made available to the multi-instance processes.

Each course attribute is independent from the others and from the rest of the system. This means that the selection of attribute values can be modified, leaving the rest of the captured details of attributes unchanged. Once the selections for a particular course attribute have been made, the next process is to validate this selected information in the Post-Processing Stage.

3. Post-Processing Stage.

   First the data and the path are validated against the existing records in the database. If duplicates are found, then this example data will not be processed further. The valid data and the path will be passed to the '*Regular expression Generator*' to generate their regular expressions. The Regular expression Generator is a tailored script written to translate the received value to a regular expression notation (see 'Automatic Regular Expression Generator' subsection below for further detail) with proper handling of whitespace characters. Both regular expressions are validated against the existing records in the database to avoid duplication and once this is clear, all the relevant details are saved temporarily. This is the last process for an attribute and the user is not allowed to do the same attribute process again.

   The same processes (Stage 2 and 3) are repeated for all available course attributes from this web page. Once this is completed, the extracted data should be ready for insertion into the target database.

   All the jQuery paths are then tokenised and compared with the stored rules in the XML file and any new tokens will be kept. Next, the value (5) in the Link_status of the current web page is changed to 3, indicating that this web page has been successfully processed. The whole process will be repeated for the next URLs.

jQuery Path Patterns

In this research jQuery is essential to define the position of the selected information on the web page, which will then be used to define one part of the extraction patterns (structural). As mentioned earlier, extraction pattern is comprised of the combination of structural and lexical patterns.

The HTML document is first made clickable. When a user clicks on information, the jQuery path elements are collected and joined together. This allows the parents, siblings and children, if available, to be trackable. To join the path elements from the selection up to the parent, a script needs to be written. For example, if the information is in a second column of the second row in a table of the second division, then the jQuery to define the absolute path of this information is html>body>div:eq(1)>table>tr:eq(1)>td:eq(1).

The absolute path performs well on a single instance of a course attribute. This means the definitive path is achievable through jQuery using the most detailed node, i.e. td:eq(1) in the above example. However, this absolute path is not applicable to multiple records extraction. Multiple records are commonly presented in tables or lists. For example, the titles of the course could be listed in the second column of row 2 to row 5 in a table. Therefore specifying the detailed node, i.e. tr:eq(1)>td:eq(1) will only locate the second column of the second row. The extraction of single and multiple records from a web page is described in the ‘Capturing and analysing Patterns’ section below.

Extraction Rules

The TS-WIE system, which adds an interactive functionality to the automatic WIE is responsible for extending the system’s extraction rules, learning from a set of training data provided by the user. The generation of a successful regular expression (wrapper) for new data depends heavily on the availability of the rule component. Thus updating the rules through new discovery from the training set in an incremental manner is necessary to accommodate a new data pattern. However, writing useful extraction rules is a difficult and tedious

task, especially as it requires:

- Regular examination of web pages for any technological/structural/content updates.
- Extensive understanding of the proper structural construction of the extraction rules, which can cope with future addition of new rules.
- Writing exhaustive rules to retrieve the important data.

Because of those reasons, this research uses predefined XML grammar structure defining the rules classification defined in Section 5.2.4. Since one of the goals is to allow addition of the rules for the new extraction task, which purpose is to reduce the (manual) pattern/wrapper generation effort, it makes sense to build an algorithm to automatically verify and extend the rules collection. While the incremental rule is mostly concerned with the data pattern (in regular expression notation) and HTML tags, other rules such as *keywords* (which rarely change) and *open tags* (e.g.'<DIV>') remain fixed.

The following section describes the process of capturing and analysing patterns for both data and path, before they are accepted into the database for future query.

Capturing and analysing Patterns

Initially, the TS-WIE system will receive data, about which it has no knowledge of the pattern, selected on the web page by the user. One of the findings in this research is that the automatic extractor fails to recognise the required data from all web pages which it has insufficient extraction rules to produce the correct patterns.

Figure 6.7 depicts the process of capturing and analysing the captured data. Once the user has made the first selection, the DOM tree (jQuery path) and the selected information are extracted. If the second information is selected, this indicates that the web page has multiple records. There are two assessors involved; Instance Filter and Rule Analyser.

Instance Filter has to ensure that the data selected for the first and the second examples are not the same value but having the same data format. As in case of *Rule Analyser* for multiple records extraction, both paths must have at least a common parent. A bottom-up approach is applied where the analysis begins with the most specific paths before they are formed as one computed general path. This means to infer a minimal regular grammar from a finite set of examples. The paths are tokenised and compared. These path expressions are then generalised to form a single path by replacing some of the tokens with wildcards (.*?), for example “<div><” the “irrelevant data” is replaced with ‘.*?’ to become <div>*.?</div><div> or dropping redundant tokens (e.g. html>body>div>table>tr>td to simply just a table>tr>td).

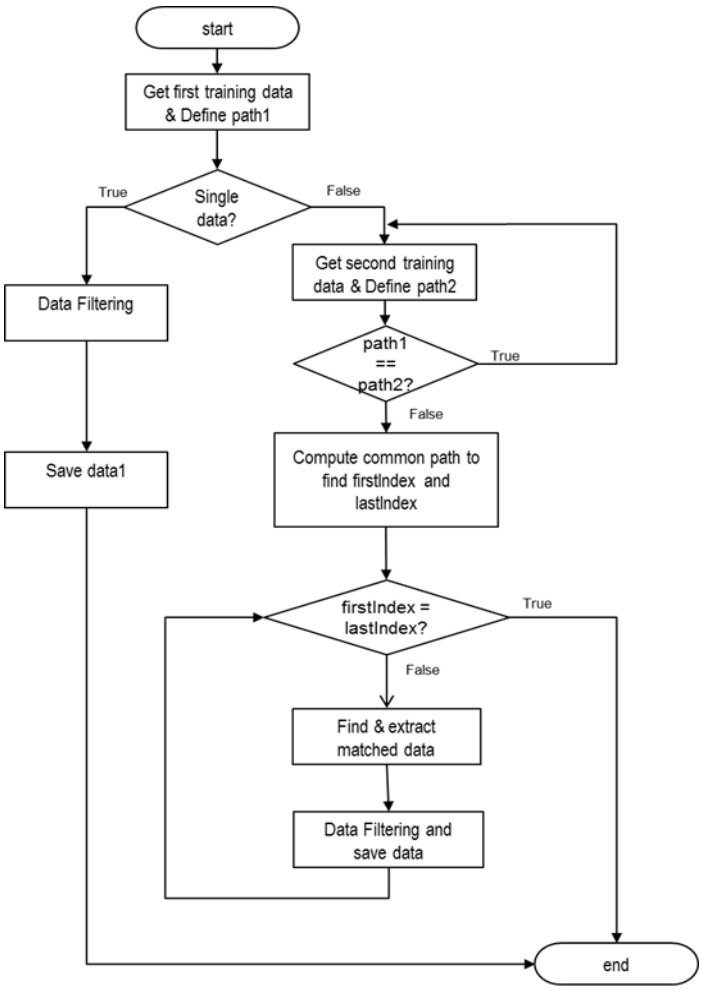


Figure 6.7 Accept Training Examples & Extraction process of the TS-WIE system.

The algorithm to produce the generalised path pattern has to ensure a balance between flexibility and specificity. The generalised path would match several similar data items and it is useful to the next process - the regular expression generator (this is explained in the next section). The path should not be too specific that it only works for one specific page such as “div:eq(1)>table.>tr:eq(1)>td:eq(1)” or too general e.g. “div table tr td” that it allows extraction of all data that it can match including the incorrect one.

The *Rule Analyser* is also responsible for providing markers for the first occurrences of the attribute that will be used for extraction process. It will also make a small assumption about the number of attributes available and define the marker for the last data. The use of these markers is to define the block, which the attribute values are likely to be found. Finally, the generalised path is used to find all occurrences of the attribute by adding the sibling paths starting from the first marker.

It is now that a jQuery pattern needs to be converted into a regular expression.

Automatic Regular expression Generator

Manual crafting of regular expressions may cause inconsistency or be partially correct, although there are several debugging tools for checking its validity, which are freely available such as RegexBuddy[4] and Regextutor[5]. To avoid this situation, it was decided that a specialised algorithm was needed to generate regular expressions from the given example and its jQuery path.

There are two distinct implementations of regular expressions generators. One translates the path and the other works with the instance of the course attribute. The algorithm for transforming the regular expression for a path concerns the correct number of open tags and the closing tags presented by the jQuery. Table 6.4 shows the equivalent regular expression to represent

[4] http//www.regexbuddy.com/. This website provides a downloadable tool for building and testing regular expressions.

[5] www.perlfect.com/articles/regextutor.shtml. This online tool offers regular expression checker, which is for PERL.

the actual extraction pattern based on a jQuery path: div:eq(1)>table>tr:eq(1)>td:eq(2)

The algorithm for matching the course attributes replicates Conrad's (2007) automatic regular expressions, who uses it for detecting spam in the email consisting of digits, hexadecimal, Top-level Domain (e.g. com, net and org), characters, day of the week and month. The algorithm used in this thesis has the addition of pre-defined regular expression for the structured words objects in the logic such as the format for the date and price, a word containing meta-character(s) and single character. This addition is necessary to avoid unnecessary overhead cost to try to build the regular expressions for a particular attribute value. The following Table 6.5 provides an example of translating the date pattern (20-08-2013) if it not already available in the database.

Table 6.4 – Example of jQuery path translation to regular expression

jQuery path Example		*Translation to Extraction pattern*		*Regular Expression equivalent*
div:eq(1)	➔	*<DIV></DIV><DIV>*	➔	*<DIV[^>]*>\s*</DIV>\s* <DIV[^>]*>\s**
table	➔	*<TABLE>*	➔	*<TABLE[^>]*>\s**
tr:eq(1)	➔	*<TR></TR><TR>*	➔	*<TR[^>]*>\s*</TR>\s* <TR[^>]*>\s**
td:eq(2)	➔	*<TD></TD><TD></TD> <TD>*	➔	*<TD[^>]*>\s*</TD>\s* <TD[^>]*>\s**
		(data_patten)		*See Table 6.5*
++		*</TD></TR></TABLE></DIV>*		*</TD>\s*</TR>\s* </TABLE>\s*</DIV>*

++ The closing of the tags is required to ensure a valid pattern.

Table 6.5 Example of text conversion to regular expression

Extraction Pattern Example		*Regular Expression equivalent*	*Description*
20	➔	*\d+*	*digit*
-	➔	*-*	*metacharacter*
08	➔	*\d+*	*digit*
-	➔	*-*	*metacharacter*
2013	➔	*\d+*	*digit*

In the translation of data value to regular expression, the notion of a *token* is used. Token is different from word as special characters are considered as tokens. The process of tokenising is applied which separates tokens by white space and takes into account punctuation. The algorithm to automatically generate the regular expression according to a set of logics and the proof-of-concept Perl source code of this logic can be found in Appendix 6.

A Pattern Filter is used in the case when two examples are provided. It is only used for analysing the DOM tree structures (jQuery paths). The output is a generalised path pattern in regular expression notation. The first implementation accepts jQuery path(s). A single path indicates a direct translation to regular expression. Two paths means further processing is needed (multi-instance extractions). Often the learned paths can be reduced to avoid redundant tags. Also often after reduction, this pattern may be the same with the previously learned and stored pattern. However, having too generalised pattern is risky as it provides opportunity for irrelevant data to be selected. This is tackled by incorporating the data pattern validation before the data are submitted to the database. The stored regular expression is important as it will be used again by the automatic WIE in an attempt to extract relevant data from 'never seen before' web pages.

The translated regular expression will be stored in the Path_Phenotype table, the field (pathRE) which can only hold 400 characters or less. However, any length is acceptable but needs to be set in the database prior to using the system. With this restriction, the generated regular expression must be within the set length, otherwise it is considered invalid. This step is necessary to avoid a long regular expression being truncated, which would cause an incorrect regular expression to be stored and thus disrupt the execution of the REGEXEV especially during the fitness test used in the Genotype to Phenotype mapping process.

Data Model

The data model (Figure 3.6 in Section 3.3.1) for the automatic WIE remains the same except for some minor amendments. Out of the seventeen tables in the database of the automatic WIE, only seven tables are relevant to the TS-WIE system. These are the CIE_Allowed_links, CIE_Course, GP_Context, GP_Locations, GP_Genomes, GP_Phenomes and GP_Genome_Phenome.

The CIE_Allowed_links provides an indicator to the system if URL needs further processing. The CIE_Course stores the extracted attributes from a particular URL and CIE_Context stores the name of course attributes to extract, i.e., title, price, location and date. The GP_Locations is updated if new location is found in the given example and this table is useful for the Location extraction (see Section 5.2.4). The GP_Genome_Phenome is a cross reference table for GP_Genomes and GP_Phenomes due to many-to-many relationship situation.

The GP_Genome and GP_Phenome tables, however, have to be redesigned to fit the requirement of this TS-WIE system. The GP_Phenome is split into two tables; and are renamed as Path_Phenotype and Data_Phenotype. This separation is essential to store two different types of phenotype; data and DOM tree path. The Path_Phenotype may not have any related record in the Genotype table as some of the phenomes are created by the TS-WIE system. However, the Genotype records must have a corresponding record in the Path_Phenotype table. The GP_Genome table now known as Genotype consists of all successfully evolved genomes. The GP_Genome_Phenome does not apply here as a genome translates to a phenome and a phenome can be the translation from different genomes (one-to-many relationship).

For clarification, in this thesis, the names of the tables are labelled as Genotype instead of Genome, and Phenotype instead of Phenome. This is because a Genome is an instance of Genotype and a Phenome is an instance of Phenotype. This means a record of the Genotype or the Phenotype in those tables is unique and not duplicated. Figure 6.8 depicts the database

components and their relationships, which are relevant to TS-WIE System.

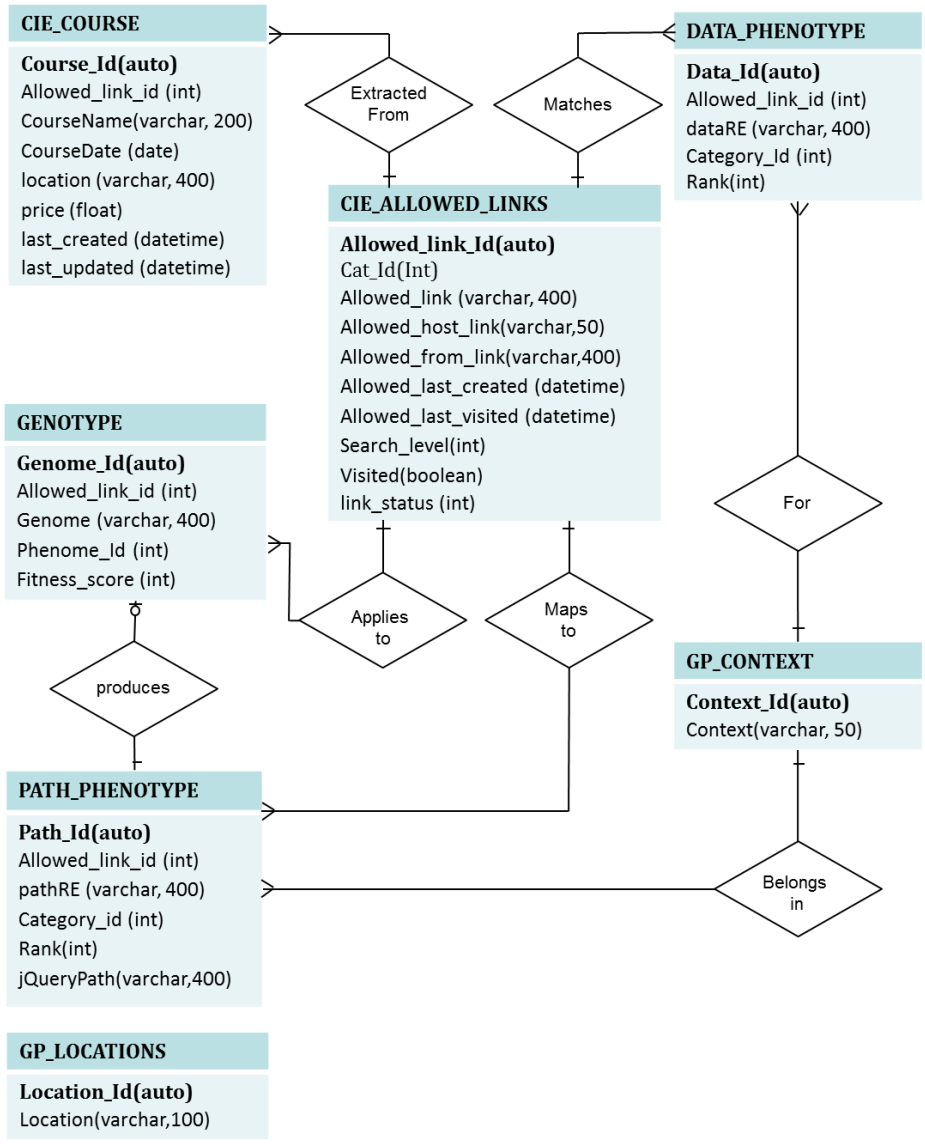


Figure 6.8 Data Model Components of TS-WIE System.

Update Database and XML

Another important function is the *Rule Incremental*. Rule Incremental automates the addition of new rules that the extractor learns, just as they are discovered on the web pages. After the *Pattern Filter* examines the data pattern, which is made up of path pattern and data pattern, the patterns (path or value) will be compared with the existing rule in the XML file and any unmatched pattern will be added in the file. Before this addition takes place,

the *Rule Incremental* will assess the newly generated pattern to determine to which rule component it should be placed. These successful patterns and rules can be reused and extended for new situations.

6.6 Experiments and Results Discussion

To investigate the effectiveness of the semi-automatic method presented in this chapter, experiments are conducted to assess the performance. According to (Sarawagi 2008), designing a model that can achieve high accuracy extraction is one of the challenging tasks facing researchers in this field.

The experiment is set to accept 'positive example(s)'. Twenty seven websites have been selected for the experiment as listed in Table 6.1. For each web page, three different standard metrics are applied; the precision, recall and F-Measure to evaluate the results of the experiments with reference to the confusion matrix (Table 2.3 in Chapter 2), which is typical for an IE system. The TS-WIE system is tested against the web page having either a single or multi-instance attributes.

Finally, this section reports the significant impact of human intervention on the yield of the TS-WIE system, which the users can "train quickly" to meet their specific needs within an acceptable level of performance. More importantly, the impact from the acquisition of the new rules in the REGEXEV experiment is reported in the following section. Figure 6.9 shows an example of extraction task from http://www.spearhead-training.co.uk/management/business-management.php and Figure 6.10 shows the extracted data.

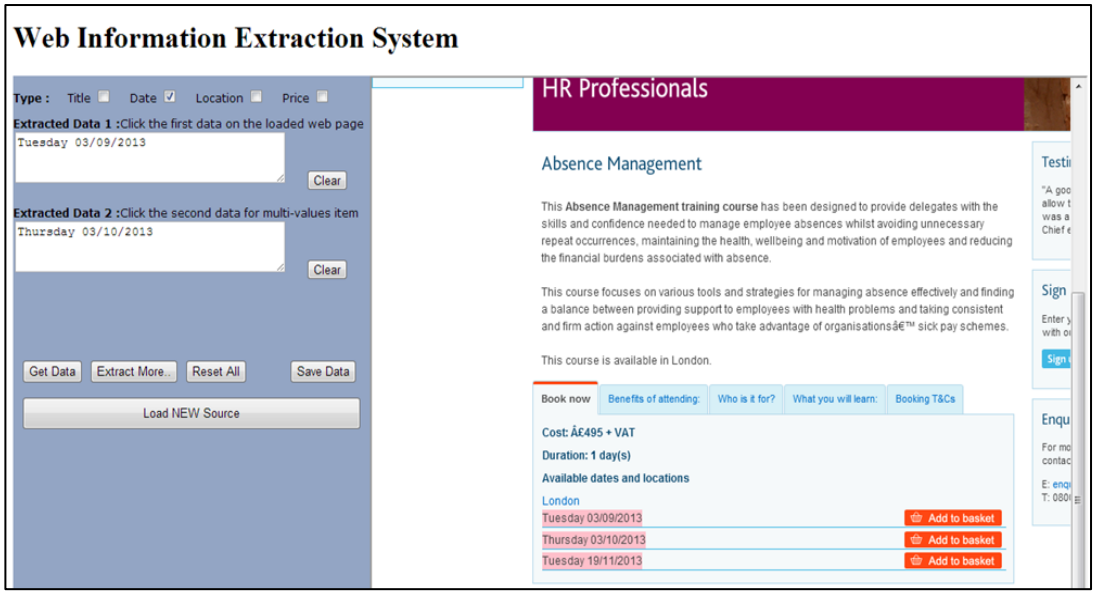
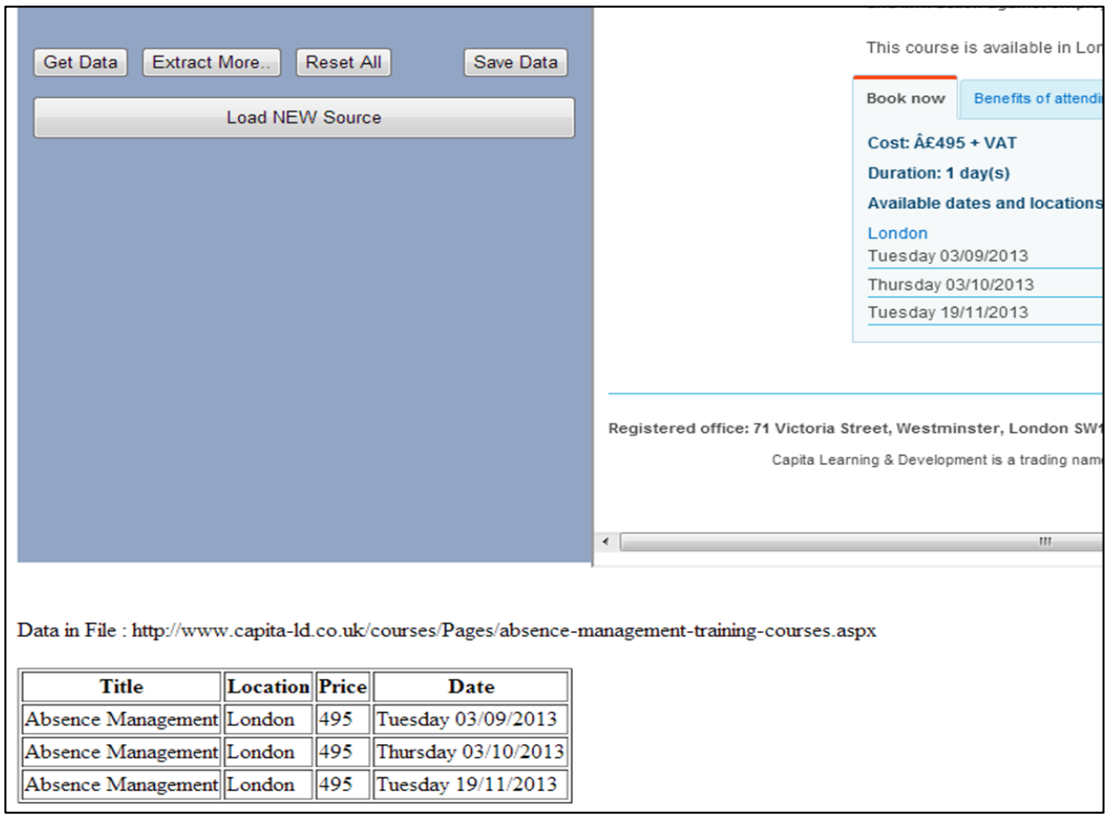


Figure 6.9. The system's response by highlighting all attribute values in pink that matches the provided examples (in multi-instance attributes web page environment).



Title	Location	Price	Date
Absence Management	London	495	Tuesday 03/09/2013
Absence Management	London	495	Thursday 03/10/2013
Absence Management	London	495	Tuesday 19/11/2013

Figure 6.10. The system's output after the task is completed. The data is stored first in the database before it is displayed back on the screen.

The following discusses the results of the experiments. It is important to note that the results are analysed and calculated as on 1st September 2013. Some

of the web pages tested in Section 5.2.4 such as underoak.co.uk and qa.co.uk were no longer alive, thus excluded from this experiment. The type of task tested for each web page varies, with some containing a single instance of an attribute while the others contain multiple instances. There are 19 out of 27 websites contain a single course title. Only 4 websites have a single date and 2 have no date at all. For the location, there are 5 websites having single venue and 6 others have none specified. Finally, the majority of the websites offer multiple course prices (depending on the location and title of course), while 5 websites offer single price and one has none specified. There are 12 websites containing multiple price instances, which offer various categories of pricing, such as members, non-members, discounted price, number of delegates, material price, international/local student and age-group price.

From the experiment result in Table 6.6, only in one of the websites (URL#25), the extractor extract all the titles. However, it also made a false extraction of one data which is not the title of the course. This is because the TS-WIE system is unable to reason the semantic of the data as a human would and the fact that these data have similar paths (generalised paths) and the same lexical pattern as the training set. In future, this could be avoided by allowing correction by the user and then add any new negative data into the irrelevant corpus and reuse this corpus to filter the extraction. However, provision of a correcting mechanism could pose further human involvement, which leads to huge effort if not handled properly.

Table 6.6 Results of experiment in % for **Title extraction**; S - Single instance; M – Multiple instances

#	URL	Type of task	Precision	Recall	F-Measure
1	www.ptp.co.uk	S	100	100	100
2	www.managementtrainingcoursesuk.co.uk	S	100	100	100
3	www.trainanddevelop.co.uk	S	100	100	100
4	www.ontargetlearning.co.uk	S	100	100	100
5	www.challengeconsulting.co.uk	S	100	100	100
6	www.itleaders.co.uk	M	100	100	100
7	www.findcourses.co.uk	M	100	100	100
8	www.campdenbri.co.uk	S	100	100	100
9	www.medicalinterviewsuk.co.uk	S	100	100	100
10	www.beauty-school.co.uk	S	100	100	100
11	academyclass.com	M	100	100	100
12	www.loucoll.ac.uk	S	100	100	100
13	www.skillsolve.co.uk	S	100	100	100
14	www.spearhead-training.co.uk	S	100	100	100
15	www.capita-ld.co.uk	S	100	100	100
16	www.cim.co.uk	S	100	100	100
17	courses.independent.co.uk	M	100	100	100
18	www.coursesplus.co.uk	M	100	100	100
19	eca.co.uk	S	100	100	100
20	register.rit.edu	M	100	100	100
21	www.cipd.co.uk	M	100	100	100
22	pgplus.bisgroup.com	S	100	100	100
23	www.hemsleyfraser.co.uk	S	100	100	100
24	www.ldl.co.uk	S	100	100	100
25	www.locksmiths-training.co.uk	M	80	100	89
26	www.theiet.org	S	100	100	100
27	www.chesterfield.ac.uk	S	100	100	100

Table 6.7 Results of experiment in % for **Date extraction**; S - Single instance; M – Multiple instances. A blank cell indicates that the web page is protected and security issue applies. A '-' cell indicates that the course attribute is not available on the web page.

#	URL	Type of task	Precision	Recall	F-Measure
1	www.ptp.co.uk	M	100	100	100
2	www.managementtrainingcoursesuk.co.uk	M	100	100	100
3	www.trainanddevelop.co.uk	M	100	100	100
4	www.ontargetlearning.co.uk	S	100	100	100
5	www.challengeconsulting.co.uk	M	100	100	100
6	www.itleaders.co.uk	M	100	100	100
7	www.findcourses.co.uk	M	100	100	100
8	www.campdenbri.co.uk	S	100	100	100
9	www.medicalinterviewsuk.co.uk	S	100	100	100
10	www.beauty-school.co.uk	M			
11	academyclass.com	M	100	10	18
12	www.loucoll.ac.uk	M	100	100	100
13	www.skillsolve.co.uk	M	100	100	100
14	www.spearhead-training.co.uk	M	100	100	100
15	www.capita-ld.co.uk	M	100	100	100
16	www.cim.co.uk	M	100	100	100
17	courses.independent.co.uk	M	100	100	100
18	www.coursesplus.co.uk	M	100	100	100
19	eca.co.uk	M	100	100	100
20	register.rit.edu	M	100	100	100
21	www.cipd.co.uk	M	100	100	100
22	pgplus.bisgroup.com	M	100	100	100
23	www.hemsleyfraser.co.uk	M	100	100	100
24	www.ldl.co.uk	M	100	100	100
25	www.locksmiths-training.co.uk	-	-	-	-
26	www.theiet.org	M	100	100	100
27	www.chesterfield.ac.uk	S	100	100	100

Due to the “different domain” security issue (https protocol), the date for URL#10 web pages in Table 6.7, which was presented in a frame, was blocked from the user, thus the content is inaccessible. In case of URL#11, each course is presented in an individual table and the tables are irregularly formatted. Multiple dates and locations are offered for each course and they are arranged in such a way that the rows represent the location and the columns represent the range of course dates (i.e. day range in the form of dd-dd e.g. 12-15) for that particular location. The months are placed as the table header. If a course is available, the date range is placed in the cell corresponding to the month and the location. If a course is not offered in a particular month for a particular location, the cell is left blank. This presentation is very complex for the algorithm to compute the relevance of the data; the task of information extraction would become almost infeasible.

Reflecting on other websites from Table 6.7, the system performs well on several web pages of similar structure such as URL#1, URL#2 and URL#7, where course records are grouped according to their course locations in separate tables. This is because the structures of these tables are regular, where the attributes are consistently arranged in specific columns, thus easier to identify.

Table 6.8 Results of experiment in % for **Location extraction**; S - Single instance; M – Multiple instances. A '-' indicates the course attribute is not available on the page.

#	URL	Type of task	Precision	Recall	F-Measure
1	www.ptp.co.uk	M	100	100	100
2	www.managementtrainingcoursesuk.co.uk	M	100	100	100
3	www.trainanddevelop.co.uk	M	100	100	100
4	www.ontargetlearning.co.uk	S	100	100	100
5	www.challengeconsulting.co.uk	-	-	-	-
6	www.itleaders.co.uk	-	-	-	-
7	www.findcourses.co.uk	M	100	100	100
8	www.campdenbri.co.uk	S	100	100	100
9	www.medicalinterviewsuk.co.uk	S	100	100	100
10	www.beauty-school.co.uk	-	-	-	-
11	academyclass.com	M	66	11	19
12	www.loucoll.ac.uk	-	-	-	-
13	www.skillsolve.co.uk	S	100	100	100
14	www.spearhead-training.co.uk	M	100	100	100
15	www.capita-ld.co.uk	M	100	100	100
16	www.cim.co.uk	M	100	100	100
17	courses.independent.co.uk	M	100	100	100
18	www.coursesplus.co.uk	M	100	100	100
19	eca.co.uk	M	100	100	100
20	register.rit.edu	M	100	100	100
21	www.cipd.co.uk	M	100	100	100
22	pgplus.bisgroup.com	M	100	100	100
23	www.hemsleyfraser.co.uk	M	100	100	100
24	www.ldl.co.uk	M	100	100	100
25	www.locksmiths-training.co.uk	-	-	-	-
26	www.theiet.org	M	100	100	100
27	www.chesterfield.ac.uk	S	100	100	100

The same security issue as the date extraction is observed in the price extraction from URL#10 web pages on Table 6.9.

Table 6.9 Results of experiment in % for **Price extraction**; S - Single instance; M – Multiple instances. A '-' indicates that the specific course attribute is not available on the web page.

#	URL	Type of task	Precision	Recall	F-Measure
1	www.ptp.co.uk	M	100	100	100
2	www.managementtrainingcoursesuk.co.uk	M	100	100	100
3	www.trainanddevelop.co.uk	M	100	100	100
4	www.ontargetlearning.co.uk	S	100	100	100
5	www.challengeconsulting.co.uk	M	100	100	100
6	www.itleaders.co.uk	M	100	100	100
7	www.findcourses.co.uk	M	100	100	100
8	www.campdenbri.co.uk	M	100	100	100
9	www.medicalinterviewsuk.co.uk	S	100	100	100
10	www.beauty-school.co.uk	M			
11	academyclass.com	M	100	100	100
12	www.loucoll.ac.uk	M	100	50	66.7
13	www.skillsolve.co.uk	S	100	100	100
14	www.spearhead-training.co.uk	M	100	100	100
15	www.capita-ld.co.uk	S	100	100	100
16	www.cim.co.uk	S	100	100	100
17	courses.independent.co.uk	M	100	100	100
18	www.coursesplus.co.uk	M	100	100	100
19	eca.co.uk	M	100	100	100
20	register.rit.edu	-	-	-	-
21	www.cipd.co.uk	M	100	100	100
22	pgplus.bisgroup.com	M	100	100	100
23	www.hemsleyfraser.co.uk	M	100	100	100
24	www.ldl.co.uk	M	100	100	100
25	www.locksmiths-training.co.uk	M	100	100	100
26	www.theiet.org	M	100	100	100
27	www.chesterfield.ac.uk	M	100	100	100

In summary, based on the results of the experiments, in which the web pages allow access, the TS-WIE system achieved the following result in Table 6.10:

Table 6.10 An average performance in % of the TS-WIE system.

Attribute	Precision	Recall	F-Measure
Title	95.6	100	99.3
Date	100	77.5	79.5
Location	95.1	77.8	80
Price	100	88	91.6

The experiments show that this system works perfectly well on single instances of each course attributes and properly structured multiple records for all web pages. Observing all the results presented above, the system performs poorly only on the following three complex cases:

1. Irregularly structured data, especially <table> presentation, where nesting and cell span are involved. In this case, two examples are insufficient to inform the system that the attributes are distributed in two or more tables and each table has various cells structures to present all instances of an attribute.

2. Dissimilar underlying format of data provided by the user.

3. Data are in the drop down list, thus the selection points to the same path. This information is insufficient to suggest the existence of multi-instance tasks.

A crude solution to handle all the three cases above would be to extract course attributes intended by providing enough samples (Carlson et al. 2010) to generate reliable patterns. However, determining how many is enough is not a straightforward task and a very large number of samples expected from the user is computationally expensive. This could be reduced by designing an algorithm to discover a number of clusters in the data that it calculates correct rather than having them as input (Vlachos et al. 2009) and the user is required to verify them for further correction, rejection or confirmation. This is not possible at this time and would be one of the future works.

6.7 Challenges

In this section, five challenges have been identified during the development of the TS_WIE system.

Challenge 1 - Human assistance.

Researchers have identified the importance of 'good' examples and the amount of labelled examples to achieve high quality of extraction. However, these two criteria are difficult to achieve without affecting the performance of the system and human effort. There must be a trade-off between them to get the best result possible. The accuracy of the results depend on the input from the user, thus it is important to get the correct representative examples. The more accurate the data that are selected, the higher the success rate of the extraction. On one hand, some researchers agree that semi-supervision yields better results than fully automated, although it is necessary that the user interaction be kept at a minimum. On the other hand, human assistance helps to increase the knowledge of the automatic WIE to hit the correct data, which it currently fails to identify thus providing an opportunity for a wider span of extraction coverage.

The TS-WIE addressed this challenge by making the web page clickable and the user hovers over the required information and clicks on it to select it. This helps to reduce any typing error which normally happens through typing in the information.

Challenge 2 - Data structure/presentation.

Websites normally spread their information on multiple web pages, hyperlinked from the main web page. These web pages may present information in a similar structure but it is not guaranteed. A small change in the pattern may cause the regular expression to fail. Some of the known issues include irregular information or missing data in tables, information that spans across multiple pages, use of images, bad structure and restricted access pages. Thus, due to these discrepancies, it would be a challenge to generalise the different layout structures and come up with a

more generalised extractor solution. However, the motivation of this research is that these web pages have some kind of structure, and some share common characteristics.

A generalisation technique (to define the location of multiple instances of a course attribute) presented in this chapter helps to relax the issue. The availability of the jQuery library to find the common parents shared by these multiple instances provides the means to identify all the target siblings and/or their children.

Challenge 3 - Evolution of the Web technology.

Web pages have evolved from static to dynamic and interactive due to introduction of new technology such as JavaScript and AJAX. Adar et al. (2009) have observed that there are two types of web page changes; structural change and content change. These changes are made for various reasons ranging from updating information to reshaping. They also pointed out that the changes made on the content of the web page (amount of textual change) are much more frequent than the structure changes (DOM-level changes). From the 55,000 web pages that they observed, the content change is done as often as every 60 minutes, i.e., in the case of the plot for the New York Times homepage. This requires a robust extraction system, which can cope and evolve with such changes.

The TS-WIE system is limited to work with websites that employ JavaScript to present the course information such as www.rit.edu for public courses. This is because each detail web page URLs are only accessible through the hyperlink (URL is hidden) as well as the contents not being visible within the HTML documents. However, due to this invisibility, this URL would not be picked up by the crawler in the first place. Furthermore, the layout structure, such as tables and lists, which is most common today, will change quickly as the new styling presentation - Cascading Styling Sheet (CSS) is increasingly accepted. Therefore, a fixed knowledgebase will soon suffer and proposing a system using a dynamic knowledgebase has the advantage of eliminating a technical expert to meet this new situation.

Challenge 4 – developing a more efficient GP method

The main technical challenge is to figure out how to create a new rule based on the training set provided by the user and generalise this rule so that it has high overall coverage. The rule should not be too general that it likely captures more irrelevant information than the required attributes, nor too definite that it is only useable in a specific web page. This then led to a new challenge of how the GP method can manage these new rules and efficiently generate new regular expression patterns.

The generated extraction pattern points not only the required information but also the irrelevant content if this relevant information is part of a paragraph or a section. To decrease the severity, the data format was applied to make this separation so only the relevant information is sent to the database. Although the result may be incomplete if the data format generated from the example is insufficient to identify the complete information, however this problem cannot be completely avoided.

Challenge 5 - Adaptation to the existing system.

The Xhemali's proposed WIE system for ATM is fully automatic and uses GP to evolve its extraction rules. Ideally, once the relevant items from the crawled page are discovered and extracted, the data are transferred to the ATM's database, which could be accessed by the user through the ATM's Customer Relationship Management (CRM) software. The automatic WIE depends on its collection of extraction rules in the XML file to generate regular expressions to match the data, and these regular expressions can be reused on other web pages with similar presentation.

It is very important that the TS-WIE system can cope with this environment without disrupting the database. Direct comparison with Xhemali's Automatic WIR/WIE system is not possible for an unavoidable reason. During the duration of this research, this automatic system is not installed at Apricot Training Management server and the codes are not available due to a very serious computer crash. This is unfortunate as finding the

effect of the TS-WIE system on the automatic WIR/WIE and the evaluation on the integration of the two systems was not possible. However, it is strongly believed that the evaluation directly tests the usefulness of the TS-WIE system against the automatic WIR/WIE, thus achieving the same output if the actual evaluation would have been possible. This is because the model of the REGEXEV in this research was designed to simulate the automatic generation of regular expression in the automatic WIE.

This thesis attempted to develop a dynamic extraction pattern based on updatable extraction rules that extracts the correct and complete instances of course attributes. Having listed all the challenges encountered while developing the TS-WIE system, only Challenge 5, which is a special and unavoidable case, remains unsolved. However, Challenges 1 to 4 have been successfully addressed. The next section attempts to demonstrate the impact of improved extraction rules provided by the TS-WIE system on the performance of REGEXEV, using the same metrics and rules structure as described in Section 5.2.4.

6.8 REGEXEV experiment revisited

The existing extraction rules used by the automatic extractor are manually built and this has put a limitation to the kind of information it can extract. However, it is postulated that teaching the extractor to add new rules, which are identified from the newly discovered patterns, into its rule collection provides a wider scope of information it can extract. In this section, this hypothesis was tested and the following results were achieved. It is important to highlight that only URLs which didn't achieve 100% precision rate (7 out of 16 websites) are included to demonstrate the effect of incremented extraction rules.

Table 6.11 A repeat of REGEXEV experiment to URLs in Table 5.8 that have precision of less than 100%. The generation of regular expressions are based on the incremented extraction rules by TS-WIE system.

(a) Title extraction

#	URL	% Hits (seeds)	Performance improvement compared with previous %	Generations		
				Best	Avg	Med
1	www.itleaders.co.uk	**93.3**	6.3	0	2.8	0
2	www.campdenbri.co.uk	**100**	20	0	0.4	0

(b) Date extraction

#	URL	% Hits (seeds)	Performance improvement compared with previous %	Generations		
				Best	Avg	Med
1	www.ptp.co.uk	100	50	0	11	9
2	www.challengeconsulting.co.uk	*60*	-10	0	1.1	0
3	www.campdenbri.co.uk	**100**	20	0	16.3	9.5

(c) Location extraction

#	URL	% Hits (seeds)	Performance improvement compared with previous %	Generations		
				Best	Avg	Med
1	www.ontargetlearning.co.uk	**93.3**	86.3	0	6.2	0
2	www.challengeconsulting.co.uk	**96.7**	76.7	0	9.8	2
3	www.findcourses.co.uk	*80*	-13	0	30	13
4	www.medicalinterviewsuk.co.uk	**100**	100	0	0.3	0

(d) Price extraction

#	URL	% Hits (seeds)	Performance improvement compared with previous %	Generations		
				Best	Avg	Med
1	www.ptp.co.uk	100	37	0	8	5
2	www.itleaders.co.uk	90	17	0	13	2
3	www.findcourses.co.uk	*56.7*	-20.3	0	1	0

It is important to note that the result above is valid as at 3rd October 2013 and because the courses offered are very sensitive to the date, i.e. courses are dated beyond the current date, therefore, some of the content of the web pages changed since the last experiment reported in Section 5.2.4.

Based on the above repeat experiments, the following has been noted:

#	website	attribute	Discussions :
(i)	Itleaders.co.uk	title	The web page repeats the title in the main body as a sub title, which is presented in a smaller heading tag. Although, in a sense it is correct, this is however treated as false negative. However, this happens less often, so rather than trying to feed the system with more complicated fitness criteria, it is more feasible to use the TS-WIE system to extract it.
(ii)	Challengeconsulting.co.uk	date	False negatives were extracted which presents valid date. This date is mainly the date for the other course titles on offer. The successful achievement of rerunning the program to hit the target dropped from 70% to 60%. An improvement to this could be to focus the search in the main content, avoiding all the noise.
(iii)	Findcourses.co.uk	Location	Duplicate instances of valid attribute have affected the performance of the system especially if the web page has some promotional information of the other courses within the same website. It has been observed that adding new elements such as HTML tags to the rules collection has a drawback. Not only does it provide new opportunity to discover new information presentation within the web pages/websites, it also provides an opportunity of new search area, which may suggests irrelevant information. The main content search could solve this issue.
(iv)	Findcourses.co.uk	Price	Same observations as (iii)

Overall, the result shows that there is a significant improvement in the precision. By providing more HTML tags, it was observed that the result is more accurate, extracting the most detail rather than the whole sentence or paragraph (within the filtration criteria specified). For example, “table tr td” is more accurate than “table tr”.

As expected, these experiments have experienced a similar impact of increased search space. There is a significant increase of performance overhead required to reach a fit solution, however, surprisingly, this is not the case for some. For example, the average generation required for evolving the extraction pattern for the date in www.campdenbri.co.uk falls from 38.6 to 16.3, although the median increases from 7 to 9.5. This shows that by adding a new grammar definition, with the correct formation of extraction pattern, the system hit the right target directly, thus resulted in better performance.

Another finding is that the improvement in the performance is the result of having the unique data pattern comparisons for each attribute, which patterns were defined from the training examples provided by the TS-WIE system. As for the location, difficulty can be seen when irrelevant information mentioning valid locations on the web page existed such as statements announcing all the available locations where the courses are operating. However, having more locations (not restricted to just the name of the city) in the database yields a much more accurate result compared to just depending on the keywords (such as location, venue and held). This was demonstrated by the experiment on www.medicalinterviewsuk.co.uk.

6.9 Chapter Summary

This chapter presents the semi-automatic technique for web information extraction through the development and implementation of the TS-WIE system, which takes advantage of human supervision combined with a set of training data. It describes the main components of the TS-WIE system, which consists of three main processes; Pre-processing, attribute extraction and Post-processing. Based on the experiments, the system performed perfectly well on extracting single instance and multiple instances that appeared in regular nested structure. On the contrary, the system demonstrated poor performance on multiple instances of attribute(s) in badly structured data.

The proposed technique assesses the data selected by a human user,

defining its DOM tree structure in jQuery notation and data format. These new patterns are then analysed by breaking them down into smaller pieces to identify if new patterns exist and the XML rules are incremented accordingly. One of the key discussions is the novel systematic method of building a 'precise' regular expression pattern based on the given example. This new regular expression is useful to the automatic WIE system to discover similar pattern in newly discovered web pages, relevant to the course training domain. Also, the addition of new pieces of rules into the regular expression grammar helps to generate new patterns which could be used to extract the data that have "never seen before" structure or format.

Finally, the chapter concluded with empirical evidence demonstrating a significant precision improvement by REGEXEV using the same XML rule structure as described in Chapter 5 with incremented rule elements.

Chapter 7

Conclusions

7.1 Chapter Overview

Chapters 4, 5 and 6 extensively discussed the major work in this research. This chapter briefly draws the conclusions in relation to the aims and objectives of the study outlined in Chapter 1.

In this research a combination of approaches (fixed-block length genotype and XML rules as an external file) were applied to evolve both areas; computer programs and regular extraction patterns. Although the field of Web Information Extraction (WIE) has been extensively studied since 1990, the application of Genetic Programming (GP) with dynamic extraction grammar or rules has remained unexplored. In this thesis, the semi-automatic WIE that supports dynamic grammar for the evolution of the extraction patterns has been presented and discussed in detail. The first section of this chapter summarises the main contributions of the research, with the limitations of the study presented in the following section. Finally, the last section identifies the opportunities for further research.

7.2 Summary of the Key Contributions

The research aimed to provide a robust WIE solution that is teachable by humans at an acceptable time and human effort. The human provides a set of training examples for the system to learn the newly discovered extraction rules or tokens to provide an improved grammar and lexicon. A well-designed Web Extraction System must consider the degree of automation in relation to the quality of extraction. It is essential to ensure there is a balance between human involvement and the accuracy of the extraction. In the literature, this human involvement ranges from creating the specific wrappers (expert user)

to providing a set of training examples for the system to learn the extraction patterns (end user). Furthermore, making the patterns extensible is a novel approach, which will help to extend the capability of the extractor to cope with future changes in the relevant Web sources. In general, the generation of the extraction patterns in a semi-automatic approach is influenced by a number of correct positive examples, which may be further supplemented with negative examples.

In order to test the research hypothesis *"A Teachable Semi-automatic Web Information Extraction System (TS-WIE) with human supervision helps to achieve high quality extraction and may increase adaptability to a wider scope of domains compared to an automatic Web Information Extraction System alone"*, a prototype system for Web Information Extraction was developed. The analysis, design and implementation of the TS-WIE (prototype) system were completed before an evaluation took place to measure the efficiency of this system. The system was to generate the relevant regular extraction patterns and increment the extraction rules for future use. The evaluation was made based on the experiments of the training course domain and it shows that the approach of using extensible extraction patterns has significantly improved the precision. The key feature of this approach is the involvement of a human for augmenting a WIE system by teaching the system the new extraction rules by example. These new rules allow the WIE system to generate some new extraction patterns to promote a new discovery.

The thesis began with the review of a complete software evolution technique proposed in Withall (2003) and Xhemali (2010b). Evolution principles, such as Genetic Programming and Genetic Algorithms, can help to automate the generation of the successful extraction patterns, without requiring direct human involvement in crafting the patterns. The effectiveness of the extraction method to generate a good extraction pattern, however, depends on the knowledgebase (rules) available to it. There are some restrictions in the work of both Withall and Xhemali, which include restrictive rules and a 'repair function' to produce a complete and syntactically correct program. This would require much effort and expertise in programming to maintain them,

thus making this approach impracticable for deployment in a general business setting and they cannot be easily scaled up.

The research presented in this thesis has focused on relaxing this method by introducing a 'clean grammar' concept and optimal design. This new approach is further improved by 'bias'ing the initial population with a successfully evolved solution and implementing a modularisation concept. Manipulating initial population helps to produce better offspring than random ones, however, incorporating modularisation proved to be better. The results of this program evolution, which used structured rules, are very important to determine if it is suitable to be extended to the evolution of extraction patterns, where the rules are much more complex and less structured. Using GP to evolve the extraction patterns has not been wholly addressed by the previous work, especially when the previously processed web page changed or never seen before web pages are involved, which demand new extraction rules. The following are the key contributions of the research:

1. A successfully evolved, syntactically correct and complete program to solve a particular computer problem can be achieved by applying a 'clean grammar' and fixed block genotype without depending on a 'repairing function' (Section 4.4 demonstrated this achievement). In the previous works such as Xhemali (2010a), Withall (2003), Ryan et al. (1998) and Banzhaf (1994), this 'repairing function' has been used to ensure the validity of the generated computer program. The 'clean grammar' introduced by this research contains rules in hierarchical structure and follows the correct programming syntax construct. This eases the rule extension allowing implementation of systematic increment or modification.
2. The experiment in Section 4.4 saw the performance of GP with multi-objective fitness function reduced the generations required to half that of single-objective fitness function for the 'sorting' of lists of integers problem.
3. Another finding presented in Section 4.4 suggests that evolving programs should have less restrictive language subsets. Rigid rules means that the search space is limited and so are the solutions

produced. A new innovative method needs to be devised to improve the efficiency and performance of this evolutionary system. For the experiments shown, applying modularisation has made a dramatic impact on the performance of the evolution system compared to the manipulation of initial population using successfully evolved solution.

4. While the literature does not discuss or justify the choice of programming language to build the evolutionary program, this research shows empirically that it is an important issue. The graph in Figure 4.5 shows that the higher the memory consumption, the slower it is to complete each generation cycle if PHP is used as the base language. Therefore, a good programming language should have the following properties:
    a. Good memory management. Evolutionary programs consume a considerable amount of space as it involves much iteration. It is important that the unused memory be released to minimise the total memory usage. This also means the lower the memory used, the more likely that the program will not crash.

    b. Maintain good speed. The program should be able to complete the execution in a reasonable time and with acceptable computational effort. Because the evolution program in this research requires the solution (phenotype) to be executed and the result is used to determine the fitness of this solution, it is important that minimal time is used to compile this generated program. PHP is worse from a memory usage perspective compared to PERL, thus it should not be used for implementing such an evolutionary system. Both languages are interpreted and so compilation of the evolved program is unnecessary.
5. Several researchers such as Ferrara et al. (2012) and Laender et al. (2002) postulate that wrapper generation and maintenance is difficult, unless a human expert is available. There was lack of evidence to suggest that automatic incrementing of extraction rules in the presence of new tokens in the web pages can handle the evolution of the extraction patterns. Therefore, towards achieving the incremental rules,

the TS-WIE system allows humans to teach it using some training data and expect it, in a reasonable time, to be able to generalise well on new data and extract information from newly seen web pages. Based on real world data, the results show that TS-WIE perfectly handles extraction of single instances and multiple instances, which are regularly structured. However, it performs poorly on irregular structures, which presentation is very complex for the algorithm to compute the relevance of the data and the task of information extraction would become almost unfeasible.

6. Re-visiting REGEXEV experiments in Section 6.8 attempted to demonstrate the effect on performance of incremental extraction rules on the WIE with the GP system. The result shows that there was a significant improvement in the precision, recall and F-Measure. One difficulty can be seen when irrelevant information of the attributes e.g. promotion of other courses specifying the title, date and price, are presented together with the relevant ones on the same web page. In all other cases, however, adding new HTML tags component, allows the system to reach the information from the most detailed nodes, thus more specific information is extracted and in some cases, the successfully evolved patterns are achieved quickly.
7. No other work is known to have concentrated on providing human assistance (that would eliminate the need for an expert's involvement) to support the automatic WIE for the training courses domain. In particular, evolving the extraction patterns (in the form of regular expression notation) based on new rules presented in the set of training examples.

7.3 Limitation

Empirical evaluation in this thesis has shown the structural and lexical analyses to define and create the extraction rules dynamically provide a positive improvement in genetically evolving the extraction patterns. Moreover, this technique offers the ability to find a novel solution to extend

the extraction coverage. However, there are a number of limitations.

Firstly, the main structure of the extraction rule categories in the XML file was fixed. XML supports extensibility and it does not restrict the span of the vertical or horizontal structural formation and the set of keywords that can be created. The fixed rule structure applied in this thesis is built based on careful examination of the web pages and the regular expression principles. It would be difficult to implement a function that can automatically determine and create a new category, unless an expert is involved. However, this decision was made with the assumption that the requirement to revise or edit this structure is rare.

Secondly, the TS-WIE system was tested and the results were presented for the course training domain. The system is intended to demonstrate its generic use on other domains with distinct attributes of interest, such as an online book store and to examine the impact of this system in terms of the adaptability requirement and its extraction performance. Although, the system can be easily “tuned” by adding new learning components, however, a further analysis is required to define the structural and lexical form presented in the other domains before a method can be devised that has the ability to generalise the extraction patterns applicable to all these domains.

Thirdly, the proposed algorithms in this thesis have no support for the file type other than HTML document (e.g. PDF and word document are used to present the upcoming course information, like a leaflet) or protected web pages. Further research is required to do the necessary pre-processing task such as transforming these documents into XHTML or XML documents or providing an interface for the user to enter the authorisation key before the extraction process can be applied.

Finally, due to the limited number of training examples that can be accepted, the TS-WIE system cannot handle data in irregular tables. This suggests that two examples are insufficient to clearly describe this type of structure of the target.

7.4 Further work

Based on the findings of this research, the following provides insights for further investigation especially in problems involving dynamic web sources:

- Possible further work on improving the performance of the TS-WIE system is to define the important section in the web page, where the relevant attributes are expected to reside. This will confine the searching of relevant information within the valid space (normally the main content) and this technique has been applied and proven beneficial in content extraction WIE initiatives. Larger workspace means that the time taken to search is longer and irrelevant data such as advertisements could be recognised as valid by the system.
- The extractor based on the structural and lexical analysis developed in this thesis had extracted a small number of false positive data due to its similarity to the DOM tree pattern (path) and the data format of the training set. This could be further corrected by the user through a user interface and communicate this correction to the system as feedback. The system will store this new negative data in the 'low-relevant corpus', which can be reused to filter unwanted data in other web pages.

  On the contrary, provision of a correcting mechanism could pose further human involvement, which leads to a huge effort if not handled properly. A solution to simplify the learning process is by having enough samples in order to generate reliable patterns (Carlson et al. 2010). This, however, raises one challenge, i.e., to determine how many samples are sufficient having in mind that processing a huge number of samples is expected to be computationally expensive. However, the challenge is reduced by designing an algorithm that discovers a number of clusters in the data that it thought correct rather than having them as input (Vlachos et al. 2009) and the user role is to verify these clusters whether to correct, reject or accept them.
- The genotype plays an important role in the selection of individuals for

the initial population. Much fitter genomes will be carried forward to the next generation for reproduction. The new rule addition will disrupt the validity of the existing genomes because the translated phenomes are strictly dependent on the genes in the genome that mapped onto the extraction rules. Because the mapping uses modulus calculation, a change in the number of candidates to choose from would result in reaching a different solution from the time the genome was first processed. For example, if there are 5 options in a rule, the gene value of 10 will be mapped to the first option. An addition of 1 option will cause the gene to map to 4^{th} option. However, re-evaluating and updating the value of each gene, so that it maps to the same phenome as before resolves this. Although it is not possible to implement it at this time, the algorithm to handle this genome alteration is available in Appendix 7.

In a long run, the effectiveness of an automatic WIE will be reduced due to the evolution of the Web sources and improper prediction of 'unknown' data. Now, the new approach to semi-automatic WIE in support for the evolution of extraction patterns using Genetic Programming not only eliminates a human technical expert to maintain the extraction rules but also improves the extraction quality in real time.

References

ABOLHASSANI, M., FUHR, N. and GOVERT, N., 2003. Information Extraction and Automatic Markup for XML Documents. *In Blanken et al.*, pp 159–174.

ADAR, E., TEEVAN, J., DUMAIS, S.T. and ELSAS, J.L., 2009. The web changes everything: understanding the dynamics of web content, *Proceedings of the Second ACM International Conference on Web Search and Data Mining* 2009, pp. 282-291.

ADELBERG, B., 1998. NoDoSE: A tool for semi-automatically extracting structured and semi-structured data from text documents. *SIG- MOD Record 27(2)*, pp. 283-294.

ALAVI, M., 1984. An assessment of the prototyping approach to information systems development. *Communications of the ACM*, **27**(6), pp. 556-563.

ALFONSECA, E., RUIZ-CASADO, M., OKUMURA, M. and CASTELLS, P., 2006. Towards large-scale non-taxonomic relation extraction: Estimating the precision of rote extractors. In *Proceedings of the 2nd Workshop on Ontology Learning and Population:Bridging the Gap between Text and Knowledge*. Association for Computational Linguistics.

ALLEN, D., WILSON, T.D., 2003. Information overload: context and causes. *The New Review of Information Behaviour Research, 4*, pp. 31-44.

ALPAYDIN, E., 2010. *Introduction To Machine Learning.* 2 edn. Mit Press (MA).

ALPHONSE, E., AUBIN, S., BESSIERES, P., BISSON, G., HAMON, T., LAGARRIGUE, S., NAZARENKO, A., MANINE, A.P., NÉDELLEC, C. and VETAH, M.O.A., 2004. Event-based Information Extraction for the biomedical domain: the Caderige project, *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications* 2004, Association for Computational Linguistics, pp. 43-49.

ANDERSEN, P.M., HAYES, P.J., HUETTNER, A.K., SCHMANDT, L.M., NIRENBURG, I.B. and WEINSTEIN, S.P., 1992. Automatic extraction of facts from press releases to generate news stories, *Proceedings of the third conference on Applied natural language processing* 1992, Association for Computational Linguistics, pp. 170-177.

ANTON, T., 2005. Xpath-wrapper Induction by Generating Tree Traversal Patterns. *In LWA*, pp. 126–133.

APPELT, D., 1999. Introduction to Information Extraction, *AI Communications*, 12(3), pp.161-172.

ARASU, A. and GARCIA-MOLINA, H., 2003. Extracting structured data from web pages, *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data* 2003, pp. 348.

AROCENA, G.O. and MENDELZON, A.O., 1998. WebOQL: Restructuring documents, databases, and Webs. *Proceedings of the 14th IEEE International Conference on Data Engineering (ICDE),Orlando, Florida,* pp. 24-33.

ASHISH, N. and KNOBLOCK, C., 1997. *Semi-automatic Wrapper Generation for Internet Information Sources*. SIGMOD Record, 26(4), pp.8-15.

ATKINSON-ABUTRIDY, J., MELLISH, C. and AITKEN, S., 2004. Combining information extraction with genetic algorithms for text mining. *IEEE Intelligent Systems,* **19**(3), pp. 22-30.

ATM, 2010. *Apricot Training Management Website.* Available from: http://apricot-ltd.co.uk/ [20 October 2010].

AVISON, D.E. and FITZGERALD, G., 2006. *Information Systems Development: Methodologies, Techniques And Tools.* 4th edn. London: McGraw-Hill Higher Education.

BAKER, J.E., 1985. Adaptive selection methods for genetic algorithms, *Proceedings of the 1st International Conference on Genetic Algorithms* 1985, L. Erlbaum Associates Inc., pp. 101-111.

BANKO, M., CAFARELLA, M.J., SODERLAND, S., BROADHEAD, M. and ETZIONI, O., 2007. Open Information Extraction from the Web. *In Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 2670-2676, Hyderabad India.

BANKO, M., CAFARELLA, M.J., SODERLAND, S., BROADHEAD, M. and ETZIONI, O., 2009. *Open information extraction for the web,* University of Washington.

BANZHAF, W., 1994. Genotype-Phenotype-Mapping and Neutral Variation: A case study in Genetic Programming. *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature,* pp. 322-332.

BARBOSA, D., WANG, H. and YU, C., 2013. Shallow Information Extraction for the Knowledge Web. *In Proceedings of IEEE 29th International Conference on Data Engineering*, Brisbane, pp.1264-1267.

BARRERO, D., GONZÁLEZ, A., R-MORENO, M. and CAMACHO, D., 2010. Variable Length-Based Genetic Representation to Automatically Evolve Wrappers. *Trends in Practical Applications of Agents and Multiagent Systems,* pp. 371-378.

BARRERO, D.F., CAMACHO, D. and R-MORENO, M.D., 2009. Automatic Web Data Extraction based on Genetic Algorithms and Regular Expressions. *Data Mining and Multi-agent Integration*, , pp. 143-154.

BARTOLI, A., DAVANZO, G., DE LORENZO, A., MAURI, M., MEDVET, E. and SORIO, E., 2012. Automatic generation of regular expressions from examples with genetic programming. *In Proceedings of the fourteenth International Conference on Genetic and Evolutionary Computation Conference Companion*, pp. 1477-1478.

BAUMANN, S., MALBURG, M. H., HEIN, H. G., HOCH, R., KIENINGER, T. and KUHN, N., 1995. Document analysis at DFKI.-Part 2: Information extraction. *Deutsches Forschungszentrum fur Kunstliche Intelligenz, RR-95-03*.

BAUMGARTNER, R., CERESENA, M. and LEDERMULLER. G., 2005. Deepweb Navigation in Web Data Extraction. *In Proc. International Conference on Computational Intelligence for Modelling, Control and Automation*, Washington, DC, USA, pp. 698-703.

BAUMGARTNER, R., GATTERBAUER, W. and GOTTLOB, G., 2009. *Web Data Extraction Systems.* Encyclopedia of Database Systems, Springer, pp. 3465-3471.

BELAZZOUGUI, D. and RAFFINOT, M., 2013. Approximate regular expression matching with multi-strings. *Journal of Discrete Algorithms*, 18, pp 14-21.

BHIDE, M., GUPTA, A., GUPTA, R., ROY, P., MOHANIA, MK. and ICHHAPORIA, Z., 2007. Liptus: Associating structured and unstructured information in a banking environment, *SIGMOD Conference*, pp. 915–924.

BLICKLE, T. and THIELE, L., 1996. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary computation*, **4**(4), pp. 361-394.

BOEHM, B.W., 1988. A spiral model of software development and enhancement. *Computer*, **21**(5), pp. 61-72.

BOOCH, G., 1982. Object-Oriented Design. *ACM SIGAda Ada Letters*, **1**(3), pp. 64-76.

BRANK, J., GROBELNIK, M. and MLADENIC, D., 2005. A survey of ontology evaluation techniques, *Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005)* 2005, pp. 166-170.

BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C.M., MALER, E. and YERGEAU EDS, F., 2008. *eXtensible Markup Language (XML)* [Homepage of W3C]. Available from: http://www.w3.org/XML/Core/#Publications [7 October 2011].

BRAZMA, A. and CERANS, K., 1994. Efficient Learning of Regular Expressions From Good Examples. *In Proeedings of fourth International Workshop on Analogical and Inductive Inference, Lecture Notes in Artificial Intelligence*, 872, Springer, Berlin, pp. 76-90.

BREUEL, T. M., 2003. High performance document layout analysis. In *Proceedings Symp. Document Image Understanding Technology*.

BRIN, S., 1998. Extracting Patterns and Relations from the World Wide Web. In Proceedings of the International Workshop on the Web and Databases.

BRINDLE, A., 1981. Genetic algorithms for function optimization. *Doctoral dissertation, University of Alberta* .

BUITELAAR, B.P., CIMIANO, P., MAGNINI, B. and SABOU, M., 2005. Learning web service ontologies: an automatic extraction method and its evaluation. *Ontology learning from text: methods, evaluation and applications*, 123, pp. 125.

BUITELAAR, P., CIMIANO, P., RACIOPPA, S. and SIEGEL, M., 2006. Ontology-based information extraction with soba, *Proceedings of the International Conference on Language Resources and Evaluation*, pp. 2321–2324.

BURSTEIN, F.V., 2000. Chapter 7 of Research Methods for Students and Professionals Information Management and Systems. *System Development in Information Systems Research.* Centre for Information Studies Charles Sturt Uni Wagga Wagga NSW., pp. 147-158.

BUYKO, E., WERMTER, J., POPRAT, M. and HAHN, U., 2006. Automatically adapting an NLP core engine to the biology domain, *Proceedings of the Joint BioLINK-Bio-Ontologies Meeting. A Joint Meeting of the ISMB Special Interest Group on Bio-Ontologies and the BioLINK Special Interest Group on Text Data Mining in Association with ISMB*, pp. 65-68.

CAI, D., YU, S., WEN, JR and MA, WY., 2003. Extracting content structure for web pages based on visual representation. In X. Zhou, Y. Zhang, and M. E. Orlowska, editors, APWeb, volume 2642 of LNCS, pages 406-417. Springer.

CAREY, J., 1990. Prototyping: alternative systems development methodology. *Information and Software Technology,* **32**(2), pp. 119-126.

CARLSON, A. 2010. Coupled Semi-Supervised Learning. PhD Thesis, Carnegie Mellon University, Pittsburgh.

CARLSON, A., BETTERIDGE, J., WANG, R.C., HRUSCHKA, E. and MITCHELL, T.M., 2010. Coupled Semi-Supervised Learning For Information Extraction. In Proceedings of ACM International Conference on Web Search and Data Mining, New York City, NY USA.

CASTILLO, F., KORDON, A., SWEENEY, J. and ZIRK, W., 2005. Using genetic programming in industrial statistical model building. *Genetic programming theory and practice II,* , pp. 31-48.

CAVANA, R.Y., DELAHAYE, B.L. & SEKARAN, U., 2001. *Applied Business Research: Qualitative and Quantitative Methods.* Sydney John Wiley and Son.

CETINKAYA, A., 2007. Regular Expression Generation Through Grammatical Evolution, Proceedings of the 2007 GECCO Conference Companion on Genetic and Evolutionary Computation, pp. 2643-2646.

CHANDRASEKARAN, B., JOSEPHSON, J.R. and BENJAMINS, V.R., 1999. What Are Ontologies, and Why Do We Need Them? *IEEE Intelligent Systems,* **14**(1), pp. 20-26.

CHANG, C.H. and KUO, S.C., 2004. OLERA: A semi-supervised approach for Web data extraction with visual support. *EEE Intelligent Systems, 19(6),* pp. 56-64.

CHANG, C.H., HSU, C.N. and LUI, S.C., 2003. Automatic information extraction from semi-structured web pages by pattern discovery. *Decision Support Systems,* **35**(1), pp. 129-147.

CHANG, C.H., KAYED, M., GIRGIS, M.R. and SHAALAN, K.F., 2006. A survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering,* **18**(10), pp. 1411-1428.

CHEN, H., CHAU, M. and Zeng, D., 2002. Ci Spider: A Tool For Competitive Intelligence on the Web. *Decision Support Systems*, 34, pp. 1-17.

CHIDLOVSKII, B., BORGHOFF, U. and CHEVALIER, P., 1997. Towards sophisticated wrapping of web-based information repositories, *In Proceedings of 5th International RIAO Conference*, pp. 123.

CHINCHOR, N. and SUNDHEIM, B., 1993. MUC-5 evaluation metrics, *Proceedings of the 5th conference on Message understanding,* Association for Computational Linguistics, pp. 69-78.

CHOOCHAIWATTANA, W., 2012. An Algorithm of Product Information Extraction from Web Pages: a Document Object Model Analysis Approach. *In Proceedings of 2nd International Conference on Information Communication and Management*, pp. 103-107.

CIMIANO, P., and VOLKER, J., 2005. Towards Large-scale, Open-domain and Ontology-based Named Entity Classification. *In Proc. Conference on Recent Advances in Natural Language Processing.*

CIMIANO, P., HAASE, P., HEROLD, M., MANTEL, M. and BUITELAAR, P., 2007. LexOnto: A model for ontology lexicons for ontology-based NLP, *Proceedings of the OntoLex07 Workshop held in conjunction with ISWC* 2007.

CIRAVEGNA, D. 2001. Adaptive Information Extraction From Text by Rule Induction and Generalisation. In *Proceedings 17th International Joint Conference on Artificial Intelligence.* Seattle.

CLARKE, M., HINDE, C.J., WITHALL, M.S., JACKSON, T., PHILLIPS, I.W., BROWN, S. and WATSON, R., 2010. Allocating railway platforms using a genetic algorithm. *Research and Development in Intelligent Systems XXVI,* pp. 421-434.

CMMS, 2008. *Selecting a Development Approach* [Homepage of Centers for Medicare and Medicaid Services]. Available from: http://www.cms.gov/SystemLifecycleFramework/ Downloads/SelectingDevelopmentApproach.pdf [7 October 2011].

COHEN, L., MANION, L. & MORRISON, K., 2007. *Research Methods in Education.* 6th edition. Routledge, Taylor and Francis Group, London, New York.

COHEN, W. & McCALLUM, A. 2003a. Information extraction from the world wide web, KDD-03.

COHEN, W.W., HURST, M. and JENSEN, L.S., 2003b. *Web Document Analysis: Challenges and Opportunities, Chapter. A Flexible Learning System for Wrapping Tables and Lists in HTML Documents*, World Scientific.

COHEN, W.W., RAVIKUMAR, P. and Fienberg, S.E., 2003c. A comparison of string distance metrics for name-matching tasks. *In KDD Workshop on Data Cleaning and Object Consolidation*, 3, pp. 73-78).

COMSCORE INC., 2010. *comScore Reports Global Search Market Growth of 46 Percent in 2009.* Available from: http://ww.comscore.com/Insights/Press_Releases/2010/1/ Global_Search_Market_Grows_46_Percent_in_2009 [4, 2013]

CONRAD, E., 2007. Detecting Spam with Genetic Regular Expressions. *SANS Institute InfoSec Reading Room*.

COOKE, J., 1998. *Constructing Correct Software: The Basics*. Springer-Verlag.

COOLEY, R., MOBASHER, B. and SRIVASTAVA, J., 1997. Web Mining: Information and Pattern Discovery on the World Wide Web, In Proceedings of International Conference on Tools with Artificial Intelligence, pp. 558-567.

COWIE, J. and LEHNERT, W., 1996. Information extraction. *Communications of the ACM*, **39**(1), pp. 80-91.

CRAMER, N.L., 1985. A representation for the adaptive generation of simple sequential programs, J. GREFENSTETTE, ed. In: *Proceedings of the First International Conference on Genetic Algorithms* 1985, Lawrence Erlbaum Associates Inc., Mahwah, NJ, USA, pp. 183-187.

CRESCENZI, V. and MECCA, G., 1998. Grammars have exceptions. *Information Systems*, 23(8), pp. 539-565.

CRESCENZI, V., MECCA, G. and MERIALDO, P., 2001. Roadrunner: Towards automatic data extraction from large web sites, *Proceedings of the International Conference on Very Large Data Bases*, pp. 109-118.

CRESPO, A., JANNINK, J., NEUHOLD, E., RYS, M. and STUDER, R., 1994. A survey of semi-automatic extraction and transformation.

CROSSAN, F. 2003. Research philosophy: towards an understanding. *Nurse Researcher,* 11 (1), pp. 46-55.

CUNNINGHAM, D.H., MAYNARD, D.D., BONTCHEVA, D.K. and TABLAN, M.V., 2002. GATE: A framework and graphical development environment for robust NLP tools and applications, *In Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)* .

CUNNINGHAM, H. (2005). Information Extraction, Automatic. *Encyclopedia of Language and Linguistics*, pp. 665-677.

CUTLER, M., ShHIH, T. and MENG, Y., 1997. Using the Structure of HTML Documents to Improve Retrieval. *In Proceedings of the USENIX Symposium on Internet Technologies and Systems,* pp. 241-251.

DAWSON, C.W., 2009. *Projects in computing and information systems a student's guide.* 2 edn. Harlow: Addison-Wesley.

DE JONG, K.A., 1975. *Analysis of the behavior of a class of genetic adaptive systems.* Doctoral dissertation, University of Michigan.

DOAN, A.H., NAUGHTON, J.F., RAMAKRISHNAN, R., BAID, A., CHAI, X., CHEN, F., CHEN, T., CHU, E., DEROSE, P., and GAO B., 2008. Information extraction challenges in managing unstructured data. *ACM SIGMOD Record*, 37(4), pp. 14-20.

DONTCHEVA, M., DRUCKER, S., SSALESIN D. and COHEN M. F., 2007. Changes in Webpage Structure over Time, Technical Report TR2007-04-02, UW, CSE

DORSEY, P., 2000. *Top 10 reasons why systems projects fail.* Available from: http://www.dulcian.com/papers [12 December 2011].

EIKVIL, L., 1999. Information extraction from World Wide Web - A survey. Technical Report, pp. 945.

EMBLEY, D.W., 2004. Toward semantic understanding: an approach based on information extraction ontologies, *Proceedings of the 15th Australasian Database Conference-Volume 27,* Australian Computer Society, Inc., pp. 3-12.

ESTIEVENART, F., MEURISSE, JR., HAINAUT, JL. and THIRAN, P., 2006. Semi-automated extraction of targeted data from web pages. *In 22nd International Conference on Data Engineering Workshop,* pp.5.

ETZIONI, O., DOORENBOS, B. and WELD, D., 1997. A scalable comparison shopping agent for the world-wide web, *In Proceedings of the International Conference on Autonomous Agents.*

ETZIONI, O., CAFARELLA, M., DOWNEY, D., POPESCU, A.M., SHAKED, T., SODERLAND, S., WELD, D.S. and YATES, A., 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence,* **165**(1), pp. 91-134.

FERRARA, E., De MEO, P., FIUMARA, G. and BAUMGARTNER, R., 2012. Web Data Extraction, Applications and Techniques: a Survey. *arXiv preprint arXiv:1207.0246*.

FELDMAN, R., AUMANN, Y., FINLELSTEIN-LANDAU, M., HURVITZ, E., REGEV, Y., and YAROSHEVICH, A., 2002. A Comparative Study of Information Extraction Strategies, *Computational Linguistics and Intelligent Text Processing*, Springer, pp. 349-359.

FELDMAN, S., 2004. *The high cost of not finding information*. KM World, 13

FELLOWS, R. and LIU, A., 2003. *Research Methods for Construction*. Oxford: Blackwell Publishing.

FERNANDEZ-VILLAMOR, J. I., IGLESIAS, C. A. and GARIJO, M., 2012. First-Order Logic Rule Induction For Information Extraction in Web Resources. *International Journal on Artificial Intelligence Tools*, *21*(06).

FIUMARA, G., 2007. Automatic Information Extraction from Web Sources: A Survey. *In Proceedings of the Workshop between Ontologies and Folksonomies (BOF),* Michigan USA.

FLEJTER, D. 2011, *Semi-Automatic Web Information Extraction*, PhD Thesis, Poznan University of Economics.

FLOYD, C., 1984. A systematic look at prototyping. *Approaches to prototyping*, **1**, pp. 1-18.

FREITAG, D., 1998. Information extraction from HTML: Application of a general machine learning approach, *Proceedings of the National Conference on Artificial Intelligence*, JOHN WILEY & SONS LTD, pp. 517-523.

FRIEDL, J., 2006. *Mastering regular expressions.* 3 edn. O'Reilly Media, Inc.

(GAC) GLOBAL AGENDA COUNCIL ON EMERGING TECHNOLOGIES, 2012. *World Economic Forum: The top 10 emerging technologies for 2012.* Available from:http://forumblog.org/2012/02/the-2012-top-10-emerging-technologies/ [10 May 2013].

GALLIERS R., 1992. *Information Systems Research: Issues, Methods and Practical Guidelines*. Alfred Waller Ltd., Oxford, Blackwell Scientific.

GAO, X., and SINGH, M., 2013. Mining Contracts for Business Events and Temporal Constraints in Service Engagements. *IEEE Transactions on Services Computing*, 1.

GATTERBAUER, W. and BOHUNSKY, P., 2006. Table Extraction Using Spatial Reasoning on the CSS2 Visual Box Model. *In Proc. 21st National Conference on Artificial Intelligence*, pp. 1313-1318.

GATTERBAUER, W., BOHUNSKY, P., HERZOG, M., KRUPL, B. and POLLAK, B., 2007. Towards Domain-Independent Information Extraction from Web Tables

GHANI R., PROBST, K., LIU, Y., KREMA, M. and FANO, A., 2006. Text mining for product attribute extraction, *SIGKDD Explorations Newsletter*, 8, pp. 41–48.

GIBSON, D., PUNERA, K. & TOMKINS, A., 2005. The volume and evolution of web page templates, *In Proceedings of the 14th International Conference on World Wide Web*, pp. 830.

GONZÁLEZ, A., BARRERO, D., CAMACHO, D. and R-MORENO, M., 2010. A Case Study on Grammatical-Based Representation for Regular Expression Evolution. *Trends in Practical Applications of Agents and Multiagent Systems*, pp. 379-386.

GORDON, M., FAN, W. and PATHAK, P., 2006. Adaptive web search: Evolving a program that finds information. *Intelligent Systems, IEEE*, **21**(5), pp. 72-77.

GORDON, V.S. and BIEMAN, J.M., 1995. Rapid prototyping: lessons learned. *Software, IEEE*, **12**(1), pp. 85-95.

GRANLIEN, M.S., PRIES-HEJE, J. and BASKERVILLE, R., 2009. Project management strategies for prototyping breakdowns, *42nd Hawaii International Conference on System Sciences, HICSS'09, IEEE*, pp. 1-10.

GRISHMAN, R. and SUNDHEIM, B., 1996. Message understanding conference – 6: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen.

GUO, H. and NANDI, A.K., 2006. Breast cancer diagnosis using genetic programming generated feature. *Pattern Recognition*, **39**(5), pp. 980-987.

GUPTA, S., KAISER, G., NEISTADT, D. and GRIMM, P., 2003. DOM-based Content Extraction of HTML Documents, in *Proceedings of the 12th World Wide Web conference*, Budapest, Hungary.

GUYON, I., and ELISSEEFF, A., 2003. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3, pp. 1157-1182.

HAMMER, J., MCHUGH, J. and GARCIA-MOLINA, H., 1997a. Semistructured Data: The TSIMMIS Experience. *In: First East-European Workshop on Advances in Databases and Information Systems-ADBIS '97*.

HAMMER, J., GARCIA-MOLINA, H., CHO, J., ARANHA, R. and CRESPO, A., 1997b. Extracting Semistructured Information from the Web. *In Proceedings of the Workshop on Management of Semistructured Data*, pp. 18–25.

HARPER, R. and BLAIR, A., 2006. Dynamically Defined Functions In Grammatical Evolution, *IEEE Congress on Evolutionary Computation, CEC 2006*, pp. 2638-2645.

HEMBERG, E., O'NEILL, M. and BRABAZON, A., 2009. An investigation into automatically defined function representations in grammatical evolution, *15th International Conference on Soft Computing*.

HOBBS, J.R. and Riloff, E., 2010. Information Extraction in Nitin Indurkhya and Fred J. Damerau Eds: *Handbook of Natural Language Processing*, 2nd Edition, Chapman & Hall/CRC Press, Taylor & Francis Group.

HOGUE, A. and KARGER, D., 2005. Thresher: Automating the Unwrapping of Semantic Content from the World Wide Web. *Proceedings of the 14th International Conference on World Wide Web (WWW), Japan*, pp. 86-95.

HOLLAND, J.H., 1975. *Adaptation in natural and artificial systems.* Ann Arbor MI: University of Michigan Press.

HONG, J.H. and CHO, S.B., 2004. Lymphoma cancer classification using genetic programming with SNR features. *Genetic Programming*, pp. 78-88.

HOWCROFT, D. and CARROLL, J., 2000. A proposed methodology for Web development, *Proceedings of the European Conference on Information Systems*, pp. 290-297.

HSU, CN. and DUNG, MT., 1998. Generating finite-state transducers for semi-structured data extraction from the web. Information Systems, 23(9), pp.521-538.

HU, Y., XIN, G., SONG, R., HU, G., SHI, S., CAO, Y. and LI, H., 2005. Title Extraction from Bodies of HTML Documents and its Application to Web Page Retrieval. *In Proceedings of the Twenty-eighth Annual International ACM SIGIR Conference*, pp. 250-257.

HUFFMAN, S., 1996. Learning information extraction patterns from examples. *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, pp. 246-260.

HUGOSSON, J., HEMBERG, E., BRABAZON, A. and O'NEILL, M., 2007. An investigation of the mutation operator using different representations in grammatical evolution, *2nd International Symposium Advances in Artificial Intelligence and Applications, Wisla, Poland*, pp. 409–419.

IPEIROTIS, P. and Jain, A., 2008. A Quality-Aware Optimizer for Information Extraction. *Technical Report CeDER-08-02, New York University, 2007* & *ACM Transactions on Database Systems (TODS)*, 34(1), pp. 5-40

IPSOS, 2013. *Interconnected World: Shopping and Personal Finance*. Available from: http://www.ipsos-na.com/download/pr.aspx?id=11513 [20 May 2013]

IRMAK, U, and SUEL, T., 2005. Interactive wrapper generation with minimal user effort. Technical Report TR-CIS-2005-02, Polytechnic University, CIS Department.

IRMAK, U, and SUEL, T., 2006. Interactive wrapper generation with minimal user effort. In *Proceedings of the 15th international conference on World Wide Web*, pp. 553-563.

ISC, 2012. *Internet Domain Survey*. Available from: http://ftp.isc.org/www/survey/reports/current/ [1 June 2013].

JACCARD, P., 1902. *Lois de distribution florale*. Bulletin de la Socíeté Vaudoise des Sciences Naturelles, 38, pp.67-130.

JACCARD, P., 1912. The distribution of the flora in the alpine zone. *New Phytologist.* 11(2), pp. 37-50.

JESCHKE, S., NATHO, N., RITTAU, S. and WILKE, M., 2007. mArachna - Applying Natural Language Processing Techniques to Ontology Engineering. *Seventh IEEE International Conference on Advanced Learning Technologies*, pp. 571-575.

JI, H., 2010. Challenges From Information Extraction to Information Fusion, *In Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, Association for Computational Linguistics, pp. 507-515.

JI, H., FAVRE, B., LIN, W.P., GILLICK, D., HAKKANI-TUR, D., & GRISHMAN, R., 2013. Open-Domain multi-document summarization via information extraction: Challenges and prospects. In *Multi-source, Multilingual Information Extraction and Summarization* , Springer Berlin Heidelberg, pp. 177-201.

JONES, Q., RADIV, G. and RAFEALI, S., 2004. Information overload and the message dynamics of online interaction spaces: A theoretical model and empirical exploration. *Information Systems Research*, 15(2), pp. 194-210.

JONES, S. and HINDE, C., 2007. Uniform Random Crossover. *In Proceedings of the 2007 workshop on Computational Intelligence*.

JUFFINGER, A., NEIDHART, T., WEICHSELBRAUN, A., WOHLGENANNT, G., GRANITZER, M., KERN, R. and SCHARL, A., 2007. Distributed Web2.0 crawling for ontology evolution, *2nd International Conference on Digital Information Management, ICDIM '07*, pp. 615-620.

JUNDT, O. and KEULEN, M. V., 2013. Sample-based XPath Ranking for Web Information Extraction. *In Proceedings of the 8th conference of the European Society for Fuzzy Logic and Technology*.

KAISER, K. and MIKSCH, S., 2005. Information extraction - A Survey. *Technical report, Vienna University of Technology, Institute of Software Technology and Interactive Systems, Asgaard-TR-2005-6*.

KATAGIRI, H., HIRASAMA, K. and HU, J., 2000. Genetic Network Programming - Application to Intelligent Agents. *IEEE International Conference on Systems, Man and Cybernetics*, 5, pp. 3829 - 3834.

KENJIBRIEL, IM. and AKBAR, S., 2012. Analisis Pemanfaatan Metode Rule-based dan Machine Learning Untuk Ekstraksi Metadata Pada Artikel Ilmiah. *Jurnal Sarjana Institut Teknologi Bandung Bidang Teknik Elektro dan Informatika,* 1(2), pp. 280-284.

KINNEAR, K.E., 1993. Generality and difficulty in genetic programming: Evolving a sort. *In Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93,* pp. 287-294.

KLEENE, S. C. 1956. *Representation of events in nerve nets and finite automata*. In Automata Studies, Princeton University Press, Princeton N.J., pp. 3-42.

KLEIN, H. K. and MYERS, M. D., 1999. A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems. *MIS quarterly*, pp. 67-93.

KOHLSCHUETTER, C. and NEJDL, W., 2010. A Densitometric Approach to Web Page Segmentation

KOHLSCHUETTER, C., FANKHAUSER, P. and NEJDL, W., 2010. Boilerplate Detection using Shallow Text Features, *Third ACM International Conference on Web Search and Data Mining*, New York City, NY USA.

KONIG, L. and SCHMECK, H., 2009. A Completely Evolvable Genotype-Phenotype Mapping for Evolutionary Robotics, *Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO '09*, pp. 175-185.

KOZA, J. R., 1990. *Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems.* Stanford University Computer Science Department

KOZA, J.R., 1992. *Genetic Programming.* Cambridge: MA: MIT Press.

KOZA, J.R., 1994. *Genetic programming II: automatic discovery of reusable programs.* Cambridge: MIT Press, MA.

KRISHNAMURTHY, B., Gill, P. and Arlitt, M., 2008. A few chirps about twitter. *In Proceedings of the first workshop on Online social networks,* New York, NY, USA, pp. 19-24.

KRAUSHAAR, J.M. and SHIRLAND, L.E., 1985. A prototyping method for applications development by end users and information systems specialists. *MIS Quarterly*, pp. 189-197.

KRUPL, B., HERZOG, M. and GATTERBAUER, W., 2005. Using Visual Cues for Extraction of Tabular Data from Arbitrary HTML Documents. *In Proc. 14th International Conference on World Wide Web*, New York, NY, USA, pp. 1000-1001.

KUHLINS, S., and TREDWELL, R. 2003. Toolkits for generating wrappers. *In Objects, Components, Architectures, Services, and Applications for a Networked World*. Springer Berlin Heidelberg. pp. 184-198.

KUSHMERICK, N., WELD, D.S. and DOORENBOS, R.B. 1997. Wrapper induction for information extraction. In IJCAI, pp. 729-737.

KURODA, T., IWASAWA, H. and KITA, E., 2010. Application of advanced Grammatical Evolution to function prediction problem. *Advances in Engineering Software*, **41**(12), pp. 1287-1294.

LABSKÝ, M., SVÁTEK, V. and NEKVASIL, M., 2008. Information Extraction Based on Extraction Ontologies: Design, Deployment and Evaluation. *Ontology-Based Information Extraction Systems*, pp. 9.

LAENDER, A.H.F., RIBEIRO-NET0, B.A., DA SILVA, A.S. and TEIXEIRA, J.S. 2002. A brief survey of web data extraction tools, *ACM Sigmod Record*, 31(2), pp. 84-93.

LANG, H., WOHLGENANNT, G. and WEICHSELBRAUN, A., 2012. TextSweeper-A System for Content Extraction and Overview Page Detection, *International Conference on Information Resources Management*.

LAM, M., GONG, Z. and MUYEBA, M., 2008. A Method for web information extraction. *Progress in WWW Research and Development*, pp. 383-394.

LERMAN, K., MINTON, S. and KNOBLOCK, C., 2003. Wrapper maintenance: A machine learning approach. *Journal of Artificial Intelligence Research*, **18**, pp. 149-181.

LEVENSHTEIN, V.I., 1966. Binary codes capable of correcting deletions, insertions and reversals. *In Soviet physics doklady* ,10, pp. 707.

LEVERING, R. and CUTLER, M., 2006. The portrait of a common HTML web page. *In ACM symposium on Document engineering*, pp. 198-204.

LEVESQUE, S., 2002. *IBM Research - Trainable Information Extraction System* [Homepage of IBM Corporation]. Available from: http://www.research.ibm.com/IE/ [6 February, 2012].

LI, Y., KRISHNAMURTHY, R., RAGHAVAN, S., VAITHYANATHAN, S. and JAGADISH, H.V., 2008. Regular expression learning for information extraction, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp. 21-30.

LIHUI, C and LIAN, C., 2005. Using web structure and summarisation.techniques for web content mining. *Information Processing & Management,* 41(5), pp. 1225-1242.

LIU, B., CHITICARIU, L., CHU, V., JAGADISH, H.V. and REISS, F.R., 2010. Automatic rule refinement for information extraction. *Proceedings of the VLDB Endowment,* **3**(1-2), pp. 588-597.

LIU, B., GROSSMAN, R. and ZHAI, Y., 2003. Mining data records in Web pages, *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 2003, pp. 601-606.

LIU, L., PU, C. and HAN, W., 2000. XWRAP: an XML-enabled wrapper construction system for Webinformation sources, *Proceeding 16th International Conference on Data Engineering*, pp. 611-621.

LUSSIER, Y., BORLAWSKY, T., RAPPAPORT, D., LIU, Y. and FRIEDMAN, C., 2006. PhenoGO: assigning phenotypic context to gene ontology annotations with natural language processing, *Pacific Symposium on Biocomputing*, pp. 64.

MA, Q., 1995. *The Application of Genetic Algorithms to the Adaption of IIR Filters*. PhD Thesis, Loughborough University.

MAEDCHE, A., NEUMANN, G. and STAAB, S., 2003. Bootstrapping an ontology-based information extraction system. *Studies In Fuzziness And Soft Computing*, **111**, pp. 345-362.

MAKHOUL, J., KUBALA, F., SCHWARTZ, R. and WEISCHEDEL, R., 1999. Performance Measures for Information Extraction, *Proceedings of DARPA Broadcast News Workshop*, pp. 249-252.

MARTIN, J., 1991. *Rapid application development.* Macmillan Publishing Co., Inc.

MARTIN, T.P., 2005. Fuzzy sets in the fight against digital obesity, *Fuzzy Sets and Systems,* **156 (3)**, pp. 411-417.

MARTIN, T., AND SHAREF, N. M., 2011. Case studies with evolving fuzzy grammars. *In IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS),* pp. 39-45.

MATSUYA, Y., HIRASAWA, K., HU, J. and MURATA, J., 2002. Automatic Generation of Programs Using Genetic Network Programming. *In Proceedings of the 41st SICE Annual Conference*, 2, pp. 1269 - 1274.

McCALLUM, A. and JENSEN, D., 2003. A note on the unification of information extraction and data mining using conditional-probability, relational models, *In Proceedings of the IJCAI-2003 Workshop on Learning Statistical Models from Relational Data.*

MENG, X., HU, D. and LI, C., 2003. Schema-guided wrapper maintenance for web-data extraction, *Proceedings of the 5th ACM international workshop on Web information and data management*, pp. 8.

METKE-JIMENEZ, A., RAYMOND, K., and MACCOLL, I., 2011. Information extraction from web services : a comparison of Tokenisation algorithms. *In SKY2011 Workshop : Discovery and Representation of Runnable Knowledge*, Paris.

MICHELAKIS, E., KRISHNAMURTHY, R., HAAS, P.J. and VAITHYANATHAN, S., 2009. Uncertainty management in rule-based information extraction systems, *Proceedings of the 35th SIGMOD international conference on Management of data* 2009, pp. 101-114.

MILLER, J.F., THOMSON, P. and FOGARTY, T., 1997. Designing electronic circuits using evolutionary algorithms. arithmetic circuits: A case study. *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, **8**.

MILLER, J.F., 1999. An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach, *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1135-1142.

MILLER, J.F. and THOMSON, P., 2000. Cartesian genetic programming. *In: Proceedings of the Third European Conference on Genetic Programming (EuroGP2000). Lecture Notes in Computer Science,* **1802**, pp. 121-132.

MILWARD, D., BJÄRELAND, M., HAYES, W., MAXWELL, M., ÖBERG, L., TILFORD, N., THOMAS, J., HALE, R., KNIGHT, S. and BARNES, J., 2005. Ontology-based interactive information extraction from scientific abstracts. *Comparative and Functional Genomics*, **6**(1-2), pp. 67-71.

MITCHELL, T.M., 1997. Machine learning. *Burr Ridge, IL: McGraw Hill*.

MONGE, A.E. and ELKAN, C.P., 1996. The Field Matching Problem: Algorithms and Applications. *In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 267-270.

MORTON, T.S., 2000. Coreference for NLP applications, *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, pp. 173-180.

MULLER, H., KENNY, E.E. and STERNBERG, P.W., 2004. Textpresso: An Ontology-Based Information Retrieval and Extraction System for Biological Literature. *PLoS Biol*, **2**(11), pp. 309.

MUSLEA, I., MINTON, S. and KNOBLOCK, C., 1999. A hierarchical approach to wrapper induction. *Proceedings of the Third International Conference on Autonomous Agents*.

MYLLYMAKI, J. and JACKSON, J., 2002. Robust Web Data Extraction with XML Path Expressions. *Technical report. In IBM Research Report RJ 10245*.

NATIONAL ACADEMY OF SCIENCES, 1994. *Academic Careers for Experimental Computer Scientists and Engineers, National Research Council Washington, D.C*. Available from: http://books.nap.edu/html/acesc/ [20 August 2013].

NAUMANN, J.D. and JENKINS, A.M., 1982. Prototyping: the new paradigm for systems development. *MIS Quarterly*, pp. 29-44.

NEGM, N., ELKAFRAWY, P. and SALEM, A. B., 2012. A Survey of Web Information Extraction Tools. International Journal of Computer Applications(0975-8887), 43(7), pp. 19-27.

NIST, 2005. *Information Extraction Definition*. Available from: http://www-nlpir.nist.gov/related_projects/muc/ [15 May 2010].

NORMAND, E., GRANT, K., IOUP, E. and SAMPLE, J., 2009. Improving Relation Extraction by Exploiting Properties of the Target Relation. *In Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, pp. 553–561.

NUNAMAKER JR, J.F. and CHEN, M., 1990. Systems development in information systems research, *Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences*, 3, pp. 631-640.

OFFICE FOR NATIONAL STATISTICS, 2013. Internet Access - Households and Individuals, 2013. Available from http://www.ons.gov.uk/ons/dcp171778_322713.pdf [9 September 2013]

OMER, B., RUTH, B., and SHAHAR, G., 2012. A New Frequent Similar Tree Algorithm Motivated by DOM Mining Using RTDM and its new variant SiSTeR. *KDIR*, SciTePress, pp. 238-243.

O'NEILL, M. and RYAN, C., 2003. Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. *Springer-Verlag*.

O'NEILL, M. and RYAN, C., 2001. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, **5**(4), pp. 349-358.

ORLIKOWSKI, W.J. and BAROUDI, J.J. (1991). Studying information technology in organizations: Research approaches and assumptions. *Information System Research*, 2(1), pp. 1-28

PALIOURAS, G., SPYROPOULOS, C. and TSATSARONIS, G., 2011. Bootstrapping ontology evolution with multimedia information extraction. *Knowledge-driven multimedia information extraction and ontology evolution*, pp. 1-17.

PALVIA, P. and NOSEK, J.T., 1990. An empirical evaluation of system development methodologies, *Proceedings of Information Resources Management Association international conference*, pp. 72.

PHP.NET, 2010. *Garbage Collection.* Available from: http://php.net/manual/en/features.gc.performance-considerations.php [5 August 2010].

PHAM, N., WILAMOWSKI, M., 2009. IEEE Article Data Extraction From Internet. *Proceedings of International Conference on Intelligent Engineering Systems*, pp. 251-256.

PIKORSKI, J. and YANGARBER, R., 2013. Information Extraction: Past Present and Future. *Multi-source, Multilingual Information Extraction and Summarization*, Springer, pp. 23-49.

POLI, R., LANGDON, W.B. and MCPHEE, N.F., 2008. *A field guide to genetic programming.* Lulu Enterprises Uk Ltd.

POLI, R. and MCPHEE, N.F., 2009. Introduction to genetic programming, *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pp. 2775-2810.

POLI, R. and LANGDON, W.B., 2007. Genetic programming theory, *Proceedings of GECCO Conference Companion on Genetic and Evolutionary Computation*, pp. 3563-3584.

POOLE, D. and MACKWORTH, A., 2010. *Artificial Intelligence - Optimization*. Available from: http://artint.info/html/ArtInt_93.html [15 January 2013).

POPOWICH, F., 2005. Using text mining and natural language processing for health care claims processing, *SIGKDD Explorartion Newsletter*, 7, pp. 59–66.

PORTER, M. and MILLAR, V., 1985. How Information gives you competitive advantage. Havard Business Review, 63(4), pp. 149-150.

RAAVI, V. K. and SOMAYAJULA, S. P. K., 2012. Extraction Of Structured Information from Unstructured or Semi-structured Machine Readable Web Pages. *Journal of Global Research in Computer Science*,*3*(10), pp. 32-38.

RAEYMAEKERS, S. and BRUYNOOGHE, M., 2007. A hybrid approach towards wrapper induction. *In Proceedings of the Workshops Prior Conceptual Knowledge in Machine Learning and Data Mining, and Web Mining 2.0* (Berendt, B. and Mladenic, D. and Semeraro, G. and Spiliopoulou, M. and Stumme, G., eds.), pp. 161-172.

RAGGETT, D., 2012. Clean up Your Web Pages with HTML TIDY - HTMLtidy [computer software]. Available from: http://www.w3.org/People/Raggett/tidy/ [21 January 2013].

REISS, F., RAGHAVAN, S., KRISHNAMURTHY, R., ZHU, H. and VAITHYANATHAN, S., 2008. An algebraic approach to rule-based information extraction, *IEEE 24th International Conference on Data Engineering*, pp. 933-942.

RILOFF, E., 1993. Automatically constructing a dictionary for information extraction tasks, *Proceedings of the National Conference on Artificial Intelligence*, pp. 811-811.

RILOFF, E., 1996. Automatically generating extraction patterns from untagged text, *Proceedings of the National Conference on Artificial Intelligence*, pp. 1044-1049.

ROYCE, W., 1970. Managing the development of large software systems. *Proceeding of IEEE WESCON*, pp. 1-9.

RYAN, C., COLLINS, J. and NEILL, M., 1998. Grammatical evolution: Evolving programs for an arbitrary language. *Genetic Programming*, pp. 83-96.

SAIIUGUET, A. and AZAVANT, F., 2001. Building intelligent Web applications using lightweight wrappers. *Data and Knowledge Engineering*, **36**(3), pp. 283-316.

SARAWAGI, S., 2008. Information extraction. *Foundations and Trends in Databases*, **1**(3), pp. 261-377.

SCHANK, R.C., 1975. *Conceptual information processing.* New York, North Holland: Elsevier Science Inc.

SEIDLER, K. and SCHIL, A., 2011. Service-oriented information extraction, *Proceedings of the 2011 Joint EDBT/ICDT Ph. D. Workshop*, pp. 25-31.

SIAU, N.Z., HINDE, C.J. and STONE, R.G., 2012. An evolution of a complete program using XML-based grammar definition. *Proceedings of the 4th International Joint Conference on Computational Intelligence, Barcelona, Spain*, pp. 214-219.

SMITS, G., KORDON, A., VLADISLAVLEVA, K., JORDAAN, E. and KOTANCHEK, M., 2006. Variable selection in industrial datasets using pareto genetic programming. *Genetic programming theory and practice III*, pp. 79-92.

ŠNAJDER, J., BAŠIC, B.D., PETROVIC, S. and SIKIRIC, I., 2008. Evolving new lexical association measures using genetic programming. *Proceedings of the Association for Computational Linguistics. Ohio, U.S.A*, pp. 181-184.

SODERLAND, S., 1999. Learning information extraction rules for semi-structured and free text. *Journal of Machine Learning, 34(1- 3)*, pp. 233-272.

SODERLAND, S., FISHER, D., ASELTINE, J. and LEHNERT, W., 1995. CRYSTAL: Inducing a conceptual dictionary. *In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1314-1319.

SOKOLOVA, M., SHAH, M. and SZPAKOWICZ, S., 2006. Comparative analysis of text data in successful face-to-face and electronic negotiations. *Group Decision and Negotiation*, **15**(2), pp. 127-140.

STEVENSON, M. and GREENWOOD, M., 2006. Comparing Information Extraction Pattern Models. *In Proceedings of the Workshop on Information Extraction Beyond the Document.*

STOFFEL, K., and SPECTOR, L., 1996. High-Performance, Parallel, Stack-Based Genetic Programming. In Koza, J.R., Goldberg, D.E., Fogel, D.B., and Riolo, R.L. (editors) Genetic Programming 1996: Proceedings of the First Annual Conference, 224-229. Cambridge, MA: The MIT Press.

SUN, F., SONG, D. & LIAO, L., 2011. Dom based content extraction via text density, *In Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 245.

SUN MICROSYSTEMS, 2010. *Lesson: Regular Expressions.* Available from: http://java.sun.com/docs/books/tutorial/essential/regex/index.html [12 April 2010].

SUNNY, T. A. and SUNDAR, G. N., 2013. An Efficient Information Extraction Model for personal named entity. *International Journal of Computer Trends and Technology*, 4(3), pp. 446-449.

SUWA, M., SCOTT, A.C. and SHORTLIFFE, E.H., 1982. An approach to verifying completeness and consistency in a rule-based expert system. *AI Magazine*, **3**(4), pp. 16.

SVINGEN, B., 1998. Learning regular languages using genetic programming. *In Proceedings of Genetic Programming, Edited by J.R. Koza et al.*, Los Altos, CA, pp. 374-376.

SYWERDA, G., 1989. Uniform Crossover in Genetic Algorithms, *Proceedings of the third international conference on Genetic algorithms*, pp. 2-9.

TASHAKKORI, A. & TEDDLIE, C. (1998). *Mixed Methodology: Combining Qualitative and Quantitative Approaches.* Applied Social Research Methods Series, Vol. 46. Sage Publications, Thousand Oaks, London, New Delhi.

TEDMORI, S. and JACKSON, T.W., 2009. Assessing the value of an E-mail knowledge extraction system. Knowledge and Process Management, 16 (2), pp. 65-73.

TETHYS SOLUTIONS, 2010. *Automation Anywhere.* Available from: http://www.automationanywhere.com/index.htm [21 May 2010].

TIOBE, 2011. *TIOBE Community Programming Index for 2011*. Available from: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html [12 June 2011].

TOMITA, M., 1982. Dynamic Construction of Finite-state Automata From Examples Using Hill Climbing. *In Proceedings of the Fourth Annual Cognitive Science Conference,* Ann Arbor, MI, pp. 105 - 108.

TURNER, R. and EDEN, A., 2011. The Philosophy of Computer Science, *The Stanford Encyclopedia of Philosophy* (Winter 2011 Edition), Edward N. Zalta (ed.). Available from: http://plato.stanford.edu/archives/win2011/entries/computer-science.

VAN VLIET, H., 2008. *Software Engineering: Principles and Practice.* Wiley Publishing.

VARGAS-VERA, M. and MOTTA, E., 2004. AQUA–ontology-based question answering system. *MICAI 2004: Advances in Artificial Intelligence, Springer Berlin Heidelberg,* 2972, pp. 468-477.

VEENENDAAL, E.V., 1st April, 2010. *Standard glossary of terms used in Software Testing Version 2.1* [Homepage of International Software Testing Qualifications Board]. Available from: http://istqb.org/display/ISTQB/Home [7 October 2011].

VELOCITYSCAPE LTD., 2006. *Webscrapper Plus.* Available from: http://www.velocityscape.com/Products/WebScraperPlus.aspx [9 September 2010].

VLACHOS, A., BUTTERY, P., SEAGHDHA, D. O. and BRISCOE, T., 2009. Biomedical event extraction without training data. *In Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, pp. 37-40.

w3school, 2013. HTML DOM Nodes. Available from: http://www.w3schools.com/htmldom/dom_nodes.asp [13 September 2013].

WALKER, J.A. and MILLER, J.F., 2008. The automatic acquisition, evolution and reuse of modules in cartesian genetic programming. *IEEE Transactions on Evolutionary Computation,* **12**(4), pp. 397-417.

WALKER, M., 2001. *Introduction to Genetic Programming.* Available from: http://web1.cs.montana.edu/~bwall/cs580/introduction_to_gp.pdf [15 December 2011].

WILKERSON, J. and TAURITZ, D., 2010. Outlining a Practitioner's Guide to Fitness Function Design, *Proceedings of the 4th Annual ISC Research Symposium ISCRS*

WIMALASURIYA, D.C. and DOU, D., 2010. Components for information extraction: ontology-based information extractors and generic platforms, *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pp. 9-18.

WINKLER, W. E., 1990. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. *In Proceedings of the Section on Survey Research Methods*, pp. 354–359.

WITHALL, M.S., HINDE, C.J. and STONE, R.G., 2009. An improved representation for evolving programs. *Genetic Programming and Evolvable Machines,* **10**(1), pp. 37-70.

WITHALL, M., 2003. *The Evolution of Complete Software Systems,* PhD Thesis, Loughborough University, UK.

WONG T-L., 2012. Learning to Adapt Cross Language Information Extraction Wrapper. *International Journal of Appl Intell*, 36(4), pp.918–931

WORLDONE RESEARCH, 2008. *LexisNexis Workplace Productivity Survey*. Available from: http://www.lexisnexis.com/literature/pdfs/LexisNexis_Workplace_Productivity_ Survey_ 2_20_08.pdf [1 June 2013]

WU, F. and WELD, D.S., 2010. Open information extraction using wikipedia, *The Annual Meeting of the Association for Computational Linguistics (ACL-2010)*, pp. 118 - 127.

XHEMALI, D., 2010a. *Automated Retrieval and Extraction of Training Course Information from Unstructured Web Pages*, Eng.D. thesis, Loughborough University, United Kingdom.

XHEMALI, D., HINDE, C.J. and STONE, R.G., 2010b. Genetic evolution of sorting programs through a novel genotype-phenotype mapping. *Proceedings of the International Conference on Evolutionary Computation, Valencia, Spain*.

XHEMALI, D., HINDE, C.J. and STONE, R.G., 2009. Naïve Bayes vs. Decision Trees vs. Neural Networks in the Classification of Training Web Pages. *International Journal of Computer Science Issues*, **4**(1), pp. 16-23.

XU, L. and DYRESON, C., 2007. Approximate retrieval of XML data with ApproXPath. *In Proceedings of the nineteenth conference on Australasian database*,75, pp 85–96, Gold Coast, Australia.

XUE, Y., HU, Y., XIN, G., SONG, R., SHI, S., CAO, Y., LIN, CY. and LI, H., 2007. Web Page Title Extraction and Its Application. Information Processing and Management, 43, pp. 1332-1347.

YANG, Y. and ZHANG, H.J. 2001. HTML page analysis based on visual cues, *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pp. 859.

YATES, A., 2007. *Information Extraction from the Web: Techniques and Applications.* PhD
thesis, University of Washington.

ZHAI, Y. and LIU, B., 2005. Web data extraction based on partial tree alignment, *Proceedings of the 14th International Conference on World Wide Web*, pp. 76-85.

ZHANG, M., SONG, R., LIN, C., MA, L., JIANG, Z., JIN, Y., LIU, Y., ZHAO, L. and MA, S., 2002. THU at TREC 2002: Novelty, Web, and Filtering. *In Proceedings of the Eleventh Text REtrieval Conference*.

ZHU, G., BETHEA, T.J. and KRISHNA, V., 2007. Extracting relevant named entities for automated expense reimbursement, *KDD*, pp. 1004–1012.

ZHU, J., NIE, Z., LIU, X., ZHANG, B. and WEN, J., 2009. StatSnowball: a Statistical Approach to Extracting Entity Relationships. In *Proceedings of the 18th International Conference on World Wide Web*, pp. 101-110.

Appendix 1

The purpose of this appendix is to provide a basic introduction to regular expression discussed in Chapter 2. It is important for the reader to have this knowledge to understand how regular expressions can be evolved and used as an extraction pattern to extract the information of interest from the Web sources.

Regular Expression Tutorial

The notes below describes regular expression and some of the basic syntax used to create regular expression and it is taken from 'Regular Expression Tutorial - Learn How to Use and Get The Most out of Regular Expressions', available at http://www.regular-expressions.info/tutorial.html [accessed 25 December 2011].

Basically, a regular expression is a pattern describing a certain amount of text. Their name comes from the mathematical theory on which they are based. But we will not dig into that. Since most people including myself are lazy to type, you will usually find the name abbreviated to regex or regexp. I prefer regex, because it is easy to pronounce the plural "regexes". On this website, regular expressions are printed as regex. If your browser has proper support for cascading style sheets, the regex should be highlighted in red.

This first example is actually a perfectly valid regex. It is the most basic pattern, simply matching the literal text regex. A "match" is the piece of text, or sequence of bytes or characters that pattern was found to correspond to by the regex processing software. Matches are highlighted in blue on this site.

\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b is a more complex pattern. It describes a series of letters, digits, dots, underscores, percentage signs and hyphens, followed by an at sign, followed by another series of letters, digits and hyphens, finally followed by a single dot and between two and four letters. In other words: this pattern describes an email address.

With the above regular expression pattern, you can search through a text file to find email addresses, or verify if a given string looks like an email address. In this tutorial, I will use the term "string" to indicate the text that I am applying the regular expression to. I will highlight them in green. The term "string" or "character string" is used by programmers to indicate a

sequence of characters. In practice, you can use regular expressions with whatever data you can access using the application or programming language you are working with Regular expressions are text patterns, which are made up of regular notations consisting of alphanumeric characters and special characters.

Character	Usage	Example
QUANTIFIER CHARACTER		
.	Matches any one character	.ce will match ice, ace
+	Matches at least one character	+ce will match ice, ace, mice
*	Matches zero or more preceding character	.*ce will match ce, 2ce, price, dice, police
?	Matches one or none character. It will become a non-greedy quantifier; matches minimum number of times, if it immediately follows a ? , *, + or {}.	Prices? Will match price, prices
??	Preceding character is optional	prices?? Matches price, prices
*?	Matching the preceding character zero or more times.	".*?" matches "price" in "price" £200 "nett"
+?	Repeat matching the preceding character one or more times.	".+?" matches "price" and "nett" in "price" £200 "nett"
{min} {min,} {min, max}	Matches exactly the minimum of occurrences Matches at least minimum of occurrences. Matches at least minimum and not more than maximum of the preceding character.	Fe{3} will match feee but not fe, fee fe{2,} will match fee, feeee but not fe fe{1,3} will match fe, fee, feee but not feeeee.

Some examples of the Quantifier Character of Regular Expressions (substantially based on Goyvaerts, J. [6]).

[6] http://www.regular-expressions.info/reference.html

CHARACTER CLASSES		
[…]	Matches any one of the enclosed characters.	[abc] will match either a, b, or c.
[^…]	Negation – opposite of the above character. It matches any character, which is not included in the enclosed.	opposite of the above.
\	Escapes following special character and will be interpreted as normal character.	* will match a star character instead of matching zero or more occurrence of the preceding character.
^	Start of a line	^B will match 'B' in Boolean Bool.
$	End of a line	.$ will match s in occurs.
A|b	Matches either 'a' or 'b'.	Organi[s|z]ation will match 'Organisation' or 'Organization'
-	Matches a character in the range specified	[a-z0-9] matches any letter or digit – a,1, s, 8
\s , \d and \w	Matches a white space character such as line feed, a space and a tab, a digit character 0-9 and word character	\d will match 1 in abc1

Some examples of the Character Classes of Regular Expressions (substantially based on Goyvaerts, J.[4]).

The following describes how to create and use regular expressions and is taken from https://developer.mozilla.org/en/Core_JavaScript_1.5_Guide/Regular_Expressions [accessed 17 February 2012].
The examples are written in JavaScript.

Creating a Regular Expression

You construct a regular expression in one of two ways:

- Using a regular expression literal, as follows:

re = /ab+c/;

Regular expression literals provide compilation of the regular expression when the script is evaluated. When the regular expression will remain constant, use this for better performance.

- Calling the constructor function of the RegExp object, as follows:

```
var re = new RegExp("ab+c");
```

Using the constructor function provides runtime compilation of the regular expression. Use the constructor function when you know the regular expression pattern will be changing, or you don't know the pattern and are getting it from another source, such as user input.

Writing a Regular Expression Pattern

A regular expression pattern is composed of simple characters, such as /abc/, or a combination of simple and special characters, such as /ab*c/ or /Chapter (\d+)\.\d*/.

Using Simple Patterns

Simple patterns are constructed of characters for which you want to find a direct match. For example, the pattern /abc/ matches character combinations in strings only when exactly the characters 'abc' occur together and in that order. Such a match would succeed in the strings "Hi, do you know your abc's?" and "The latest airplane designs evolved from slabcraft." In both cases the match is with the substring 'abc'. There is no match in the string "Grab crab" because it does not contain the substring 'abc'.

Using Special Characters

When the search for a match requires something more than a direct match, such as finding one or more b's, or finding white space, the pattern includes special characters. For example, the pattern /ab*c/ matches any character combination in which a single 'a' is followed by zero or more 'b's (* means 0 or more occurrences of the preceding item) and then immediately followed by 'c'. In the string "cbbabbbbcdebc," the pattern matches the substring 'abbbbc'.

Using Parentheses

Parentheses around any part of the regular expression pattern cause that part of the matched substring to be remembered. Once remembered, the substring can be recalled for other use.

For example, the pattern /Chapter (\d+)\.\d*/ illustrates additional escaped and special characters and indicates that part of the pattern should be remembered. It matches precisely the characters 'Chapter ' followed by one or more numeric characters (\d means any numeric character and + means 1 or more times), followed by a decimal point (which in itself is a special character; preceding the decimal point with \ means the pattern must look for the literal character '.'), followed by any numeric character 0 or more times (\d means numeric character, * means 0 or more times). In addition, parentheses are used to remember the first matched numeric characters.

This pattern is found in "Open Chapter 4.3, paragraph 6" and '4' is remembered. The pattern is not found in "Chapter 3 and 4", because that string does not have a period after the '3'.

To match a substring without causing the matched part to be remembered, within the parentheses preface the pattern with ?:. For example, (?:\d+) matches one or more numeric characters but does not remember the matched characters.

Working with Regular Expressions

Regular expressions are used with the RegExp methods *test* and *exec* and with the String methods *match, replace, search*, and *split*. These methods are explained in detail in the JavaScript Reference.

Table 4.2 Methods that use regular expressions

Method	Description
`exec`	A `RegExp` method that executes a search for a match in a string. It returns an array of information.
`test`	A `RegExp` method that tests for a match in a string. It returns true or false.
`match`	A `String` method that executes a search for a match in a string. It returns an array of information or null on a mismatch.
`search`	A `String` method that tests for a match in a string. It returns the index of the match, or -1 if the search fails.
`replace`	A `String` method that executes a search for a match in a string, and replaces the matched substring with a replacement substring.
`split`	A `String` method that uses a regular expression or a fixed string to break a string into an array of substrings.

When you want to know whether a pattern is found in a string, use the *test* or *search* method; for more information (but slower execution) use the *exec* or *match* methods. If you use *exec* or *match* and if the match succeeds, these methods return an array and update properties of the associated regular expression object and also of the predefined regular expression object, RegExp. If the match fails, the *exec* method returns null (which converts to false).

In the following example, the script uses the exec method to find a match in a string.

```
view plain  print  ?
01.  var myRe = /d(b+)d/g;
02.  var myArray = myRe.exec("cdbbdbsbz");
```

Using Parenthesized Substring Matches

Including parentheses in a regular expression pattern causes the corresponding submatch to be remembered. For example, /a(b)c/ matches the characters 'abc' and remembers 'b'. To recall these parenthesized substring matches, use the Array elements [1], ..., [n].

The number of possible parenthesized substrings is unlimited. The returned array holds all that were found. The following examples illustrate how to use parenthesized substring matches.

Example 1.

The following script uses the replace method to switch the words in the string. For the replacement text, the script uses the $1 and $2 in the replacement to denote the first and second parenthesized substring matches.

```
view plain  print  ?
01.  <script type="text/javascript">
02.   re = /(\w+)\s(\w+)/;
03.   str = "John Smith";
04.   newstr = str.replace(re, "$2, $1");
05.   document.write(newstr);
06.  </script>
```

This prints "Smith, John".

Example 2.

Note: in the getInfo function, the exec method is called using the () shortcut notation that works in Firefox but not in most other browsers.

```
view plain  print  ?
01.  <html>
02.
03.  <script type="text/javascript">
04.   function getInfo(field){
05.     var a = /(\w+)\s(\d+)/(field.value);
06.     window.alert(a[1] + ", your age is " + a[2]);
07.   }
08.  </script>
09.
10.  Enter your first name and your age, and then press Enter.
11.
12.  <form>
13.    <input type="text" name="NameAge" onchange="getInfo(this);">
14.  </form>
15.
16.  </html>
```

Appendix 2

This appendix contains the 'Relevance' and 'Low-Relevance' words for each course attributes. These lists in the current prototype are collective words manually created after a thorough analysis of the relevant webpages and they are not exhaustive lists. In future, these lists can be automatically updated through the provision of positive and negative example and the extracted information stored in the TS-WIE database. Related words are stemmed (reduced) to their root form to avoid the analysis of the related words as independent ones. For example the word *commence, commencing* are stemmed to *commenc.*

Relevance List			
Course Title :	title	course	course name
Date :	Jan	Feb	Mar
	Apr	May	Jun
	Jul	Aug	Sept
	Oct	Nov	Dec
	Date	commenc	start
Location :	location	venue	held
Price :	price	fee	cost
	£	vat	member
	gbp		

Low – Relevance List			
Course Title :	accommodate	aim	associate
	assess	availabl	basket
	book	brochure	bury
	but	benefit	calendar
	call	cart	certificat
	center	choose	class
	click	college	contact
	cost	date	delegate
	deliver	description	discount
	drag	detail	duration
	entry	exam	equal
	enrol	fee	form
	guarantee	Home	house
	hour	improve	key
	late	link	locat

	material	my	now
	news	open	outcome
	offer	overview	outline
	price	postage	pay
	plc	pre-requisite	requisite
	programme	question	register
	relate	repeat	relevant
	site	schedule	skip
	structure	search	take
	test	touch	tutor
	this	these	type
	uk	useful	venue
	view	visit	what
	Jan	Feb	Mar
	Apr	May	Jun
	Jul	Aug	Sept
	Oct	Nov	Dec
	?	=	:
Date :	accommodate	aim	bury
	basket	brochure	calendar
	center	class	college
	contact	cost	design
	enroll	exam	enter
	find	fare	fee
	flyer	form	guarante
	hour	house	locat
	main	material	master
	our	offer	pay
	plc	print	pric
	postage	require	register
	Review	search	skip
	site	type	uk
	vat	visit	what
	while	yet	you
	?	=	
Location :	Jan	Feb	Mar
	Apr	May	Jun
	Jul	Aug	Sept
	Oct	Nov	Dec
	accomodat	about	aim
	associate	basket	book
	benefit	calendar	certificat
	call	center	choose
	class	click	contact
	cost	date	description
	duration	detail	exam
	entr	enrol	fee
	form	guarantee	house
	home	hour	link
	main	material	now
	offer	open	outcome

	pay	pric	pre-requisite
	postage	programme	register
	return	require	schedule
	site	search	skip
	visit	what	uk
	?	=	
Price :	Jan	Feb	Mar
	Apr	May	Jun
	Jul	Aug	Sept
	Oct	Nov	Dec
	accomodat	about	aim
	associate	basket	book
	benefit	calendar	certificat
	call	center	choose
	class	click	contact
	copyright	date	description
	duration	detail	exam
	entr	enrol	find
	form	guarantee	house
	home	hour	link
	main	material	multiple
	offer	iso	outcome
	pay	provider	pre-requisite
	postage	programme	require
	return	search	schedule
	site	skip	type
	visit	what	uk
	?	=	/

Appendix 3

This section contains two grammar definitions to guide the Genome to Phenome mapping in two domains discussed in detail in Chapter 4. These grammar definitions are written in XML format, validated by a specially written DTD. The first definition is useful for the evolution of a complete program and the second definition is for the evolution of regular expression pattern.

(1) A complete program grammar

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE grammar [
<!ELEMENT grammar (start,rules)>
<!ELEMENT start (nonterminal)>
<!ELEMENT rules (rule*)>
<!ELEMENT rule (nonterminal|token)*>
<!ELEMENT nonterminal EMPTY>
<!ELEMENT token (#PCDATA)>
<!ATTLIST rule
  type (selection|sequence) #REQUIRED
  name NMTOKEN #REQUIRED
>
<!ATTLIST nonterminal
  name NMTOKEN #REQUIRED
>
]>
<grammar>
  <start>
         <nonterminal name="statement" />
  </start>
  <rules>
     <rule name="statement" type="selection">
          <nonterminal name="nullstatement" />
          <nonterminal name="assignstatement" />
          <nonterminal name="ifstatement" />
          <nonterminal name="forstatement" />
          <nonterminal name="nestedforstatement" />
     </rule>
     <rule name="nullstatement" type="sequence">
          <token>;</token>
     </rule>
     <rule name="assignstatement" type="sequence">
          <nonterminal name="wvariable" />
          <token>=</token>
          <nonterminal name="rvariable" />
          <token>;</token>
     </rule>
     <rule name="ifstatement" type="sequence">
          <token>if</token>
```

```
        <token>(</token>
        <nonterminal name="rvariable" />
        <nonterminal name="operator" />
        <nonterminal name="rvariable" />
        <token>)</token>
        <token>{</token>
        <nonterminal name="statementseq" />
        <token>}</token>
  </rule>
</rules>
  <rule name="forstatement" type="sequence">
        <token>for</token>
        <nonterminal name="counter" />
        <token>(</token>
        <token>0</token>
        <token>..</token>
        <token>$#inlist</token>
        <token>)</token>
        <token>{</token>
        <token>$runtime++;  die if($runtime > $timeout);</token>
        <nonterminal name="statementseq" />
        <token>}</token>
  </rule>
  <rule name="nestedforstatement" type="sequence">
        <token>for </token>
        <nonterminal name="counter" />
        <token>(</token>
        <token>0</token>
        <token>..</token>
        <token>$#inlist</token>
        <token>)</token>
        <token>{</token>
        <token>for </token>
        <nonterminal name="counter" />
        <token>(</token>
        <nonterminal name="N1"/>
        <token>+1..</token>
        <token>$#inlist)</token>
        <token>{</token>
        <token>$runtime++;  die if($runtime > $timeout);</token>
        <nonterminal name="statementseq" />
        <token>}</token>
        <token>}</token>
  </rule>
  <rule name="statementseq" type="selection">
        <nonterminal name="statement" />
        <nonterminal name="statements" />
  </rule >
  <rule name="statements" type="sequence">
        <nonterminal name="statement" />
        <nonterminal name="statementseq" />
  </rule >
        <rule name="rvariable" type="selection">
```

```
            <token>$inlist[$tmp1%($#inlist+1)]</token>
            <token>$inlist[$tmp2%($#inlist+1)]</token>
            <token>$tmp1</token>
            <token>$tmp2</token>
            <token>$tmp3</token>
            <token>$tmp4</token>
        </rule>
        <rule name="wvariable" type="selection">
            <token>$inlist[$tmp1%($#inlist+1)]</token>
            <token>$inlist[$tmp2%($#inlist+1)]</token>
            <token>$tmp3</token>
            <token>$tmp4</token>
        </rule>
        <rule name="counter" type="selection">
            <token>$tmp1</token>
            <token>$tmp2</token>
        </rule>
        <rule name="operator" type="selection">
            <token><![CDATA[==]]></token>
            <token><![CDATA[!=]]></token>
            <token><![CDATA[>]]></token>
            <token><![CDATA[<]]></token>
            <token><![CDATA[>=]]></token>
            <token><![CDATA[<=]]></token>
        </rule>
    </grammar>
```

(2) A Regular Expression patterns grammar

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE grammar [
<!ELEMENT grammar (start,rules)>
<!ELEMENT start (nonterminal)>
<!ELEMENT rules (rule*)>
<!ELEMENT rule (nonterminal|token)*>
<!ELEMENT nonterminal EMPTY>
<!ELEMENT token (#PCDATA)>
<!ATTLIST rule
  type (selection|sequence) #REQUIRED
  name NMTOKEN #REQUIRED
>
<!ATTLIST nonterminal
  name NMTOKEN #REQUIRED
>
]>
<grammar>
  <start>
         <nonterminal name="REpattern" />
  </start>
  <rules>
      <rule name='REpattern' type='sequence'>
           <nonterminal name='opentag' />
           <nonterminal name='innercontent' />
           <nonterminal name='closetag' />
      </rule>
      <rule name='opentag' type='sequence'>
           <token><![CDATA[<]]></token>
           <nonterminal name='tagname' />
           <token><![CDATA[[^>]*>]]></token>
      </rule>
      <rule name='closetag' type='sequence'>
           <token><![CDATA[</]]></token>
           <nonterminal name='N1' />
           <token><![CDATA[>]]></token>
      </rule>
      <rule name='tagname' type='selection'>
          <token>DIV</token>
           <token>TABLE</token>
           <token>TR</token>
           <token>TD</token>
           <token>H1</token>
           <token>H2</token>
           <token>H3</token>
           <token>LI</token>
           <token>DIV</token>
           <token>P</token>
           <token>A</token>
      </rule>
      <rule name='innercontent' type='selection'>
```

```
            <nonterminal name='datacontent' />
            <nonterminal name='REpattern' />
      </rule>
      <rule name='datacontent' type='selection'>
            <nonterminal name='capturedata' />
            <nonterminal name='tag_and_value' />
      </rule>
      <rule name='tag_and_value' type='sequence'>
            <nonterminal name='opentag' />
            <nonterminal name='innertags' />
            <nonterminal name='datacontent' />
            <nonterminal name='innertags' />
            <nonterminal name='closetag' />
      </rule>
      <rule name='innertags' type='selection'>
            <nonterminal name='empty' />
            <nonterminal name='singletag' />
            <nonterminal name='singletagseq' />
      </rule>
      <rule name='singletag' type='sequence'>
            <nonterminal name='opentag' />
            <nonterminal name='closetag' />
      </rule>
      <rule name= ‘singletagseq' type='sequence'>
            <nonterminal name='singletag' />
            <nonterminal name='innertags' />
      </rule>
      <rule name='capturedata' type='sequence'>
            <token>[\s*?]</token>
            <token>(</token>
            <nonterminal name='expression' />
            <token>)</token>
      </rule>
      <rule name='expression' type='sequence'>
            <token>.*?</token>
      </rule>
  </rules>
</grammar>
```

Appendix 4

This section consists of evolved sort programs which are based on ten random seedings, from the experiments in Chapter 4. It is important to note that obviously redundant codes are removed from these code examples to provide clarity.

Evolved 'Sorting' programs

Seed 1

```
for $tmp1(0..$#inlist){
   for $tmp2($tmp1+1..$#inlist){
      if($inlist[$tmp2%($#inlist+1)]<=$inlist[$tmp1%($#inlist+1)]){
         $tmp4=$inlist[$tmp2%($#inlist+1)];
         if($inlist[$tmp2%($#inlist+1)]!=$inlist[$tmp1%($#inlist+1)]){
            if($inlist[$tmp1%($#inlist+1)]>=$inlist[$tmp1%($#inlist+1)]){
               $inlist[$tmp2%($#inlist+1)]=$inlist[$tmp1%($#inlist+1)];
               $inlist[$tmp1%($#inlist+1)]=$tmp4;
            }
         }
      }
   }
}
```

Seed 2

```
for$tmp1(0..$#inlist){
   for$tmp2(0..$#inlist){
      if($inlist[$tmp2%($#inlist+1)]>$inlist[$tmp1%($#inlist+1)]){
         $tmp3=$inlist[$tmp2%($#inlist+1)];
         if($inlist[$tmp1%($#inlist+1)]!=$tmp3){
            $inlist[$tmp2%($#inlist+1)]=$inlist[$tmp1%($#inlist+1)];
            $inlist[$tmp1%($#inlist+1)]=$tmp3;
         }
      }
   }
}
```

Seed 3

```
for $tmp2(0..$#inlist){
   for $tmp1($tmp2+1..$#inlist){
      $tmp3=$inlist[$tmp2%($#inlist+1)];
      if($inlist[$tmp2%($#inlist+1)]>=$inlist[$tmp1%($#inlist+1)]){
         $inlist[$tmp2%($#inlist+1)]=$inlist[$tmp1%($#inlist+1)];
         $inlist[$tmp1%($#inlist+1)]=$tmp3;
      }
   }
}
```

Seed 5

```
for $tmp2(0..$#inlist){
   for $tmp1($tmp2+1..$#inlist){
      if($inlist[$tmp2%($#inlist+1)]>$inlist[$tmp1%($#inlist+1)]){
         for $tmp2(0..$#inlist){
            for $tmp2($tmp2+1..$#inlist){
               $tmp4=$inlist[$tmp1%($#inlist+1)];
            }
         }
         $inlist[$tmp1%($#inlist+1)]=$inlist[$tmp2%($#inlist+1)];
         $inlist[$tmp2%($#inlist+1)]=$tmp4;
      }
   }
}
```

Seed 7

```
for $tmp2(0..$#inlist){
   for $tmp1($tmp2+1..$#inlist){
      if($inlist[$tmp2%($#inlist+1)]>=$inlist[$tmp1%($#inlist+1)]){
         $tmp3=$inlist[$tmp2%($#inlist+1)];
         if($tmp2!=$inlist[$tmp2%($#inlist+1)]){
            $inlist[$tmp2%($#inlist+1)]=$inlist[$tmp1%($#inlist+1)];
            $inlist[$tmp1%($#inlist+1)]=$tmp3;
         }
      }
   }
}
```

Seed 11

```
for $tmp2(0..$#inlist){
   for $tmp1($tmp2+1..$#inlist){
      $tmp4=$inlist[$tmp2%($#inlist+1)];
      if ($inlist[$tmp2%($#inlist+1)]> $inlist[$tmp1%($#inlist+1)]){
         $inlist[$tmp2%($#inlist+1)]=$inlist[$tmp1%($#inlist+1)];
         for $tmp2(0..$#inlist){
            for $tmp2($tmp2+1..$#inlist){
               $inlist[$tmp1%($#inlist+1)]=$tmp4;
            }
         }
      }
   }
}
```

Seed 13

```
$inlist[$tmp1%($#inlist+1)]=$inlist[$tmp2%($#inlist+1)];
for$tmp1(0..$#inlist){
   if($inlist[$tmp2%($#inlist+1)]>=$tmp2){
      for$tmp2(0..$#inlist){
         $tmp3=$inlist[$tmp2%($#inlist+1)];
         $tmp4=$inlist[$tmp1%($#inlist+1)];
         if($inlist[$tmp1%($#inlist+1)]<$tmp3){
            $inlist[$tmp2%($#inlist+1)]=$tmp4;
            $inlist[$tmp1%($#inlist+1)]=$tmp3;
         }
      }
   }
}
```

Seed 17

```
for $tmp2(0..$#inlist){
   for $tmp1($tmp2+1..$#inlist){
      if($inlist[$tmp1%($#inlist+1)]!=$tmp4){
         for $tmp1(0..$#inlist){
            for $tmp2($tmp1+1..$#inlist){
               if($inlist[$tmp1%($#inlist+1)]>=$inlist[$tmp2%($#inlist+1)]){
                  $tmp4=$inlist[$tmp1%($#inlist+1)];
                  $inlist[$tmp1%($#inlist+1)]=$inlist[$tmp2%($#inlist+1)];
                  $inlist[$tmp2%($#inlist+1)]=$tmp4;
               }
            }
         }
      }
   }
}
```

Seed 19

```
for $tmp1(0..$#inlist){
   for $tmp2($tmp1+1..$#inlist){
      if($inlist[$tmp1%($#inlist+1)]>=$inlist[$tmp1%($#inlist+1)]){
         if($inlist[$tmp2%($#inlist+1)]<$inlist[$tmp1%($#inlist+1)]){
            $tmp3=$inlist[$tmp1%($#inlist+1)];
            if($inlist[$tmp2%($#inlist+1)]>$tmp1){
               $inlist[$tmp1%($#inlist+1)]=$inlist[$tmp2%($#inlist+1)];
               for$tmp1(0..$#inlist){
                  if($tmp2==$tmp1){
                     $inlist[$tmp2%($#inlist+1)]=$tmp3;
                  }
               }
            }
         }
      }
   }
}
```

Seed 23

```
$tmp4=$inlist[$tmp2%($#inlist+1)];
for $tmp1(0..$#inlist){
   for $tmp2($tmp1+1..$#inlist){
      $tmp4=$inlist[$tmp1%($#inlist+1)];
      if($inlist[$tmp1%($#inlist+1)]>=$inlist[$tmp2%($#inlist+1)]){
         $inlist[$tmp1%($#inlist+1)]=$inlist[$tmp2%($#inlist+1)];
         $inlist[$tmp2%($#inlist+1)]=$inlist[$tmp1%($#inlist+1)];
         $inlist[$tmp2%($#inlist+1)]=$tmp4;
      }
   }
}
```

Appendix 5

This appendix contains the statistical comparison between 'solution seeded' and 'solution mutated' techniques in Chapter 4. This appendix also contains the complete result of these techniques:

i. Table 1 – solution seeded
ii. Table 2 – solution mutated

Statistical result of solution seeded and solution mutated population

The evolution system did 10 runs each with a maximum of 50,000 generations with one hundred different random seeds. The aim of this run is to find out if these two techniques of Initial Population manipulation (seeding with known solution); seeded and mutated, have any difference and if there is then to find out which technique is better.

Population size is 7 and Roulette Wheel selection was applied to select two individuals from this population to the reproduction process.

The first technique is 'seeded' or in this thesis referred to as 'Solution Seeded'. It is to have the first chromosome in the population seeded from the pool of successfully evolved solutions and the rest of the population is created at random. This technique provides a chromosome of a good solution to start with.

The second technique is 'mutated' or referred to as 'Solution Mutated' in this thesis. It takes a successfully evolved solution and applying 'one gene per chromosome mutation'. This technique assumes that the solution to the problem is not far from the existing one.

The result of these two techniques is below:

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Mutated	454.83	100	471.188	47.119
	Seeded	256.67	100	304.261	30.426

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	Mutated - Seeded	198.160	577.509	57.751	83.570	312.750	3.431	99	.001

Table 1 – Solution seeded

Seed	Gen	Time	Seed	Gen	Time	Seed	Gen	Time
1	1	0:00:00	197	613	0:00:54	491	708	0:00:56
2	155	0:00:13	199	446	0:00:36	493	95	0:00:09
3	486	0:00:36	211	42	0:00:04	497	277	0:00:25
5	151	0:00:12	223	3	0:00:01	499	51	0:00:04
7	317	0:00:24	227	1179	0:01:37	501	32	0:00:03
11	15	0:00:02	229	551	0:00:49	503	754	0:01:05
13	90	0:00:07	233	221	0:00:17	509	128	0:00:10
17	459	0:00:41	239	1102	0:01:29	511	156	0:00:14
19	353	0:00:33	241	132	0:00:12	513	334	0:00:28
23	231	0:00:18	251	436	0:00:47	521	0	0:00:00
29	0	0:00:00	257	474	0:00:43	523	152	0:00:16
31	20	0:00:02	263	262	0:00:23	541	0	0:00:01
37	1206	0:01:34	269	1197	0:01:44	547	457	0:00:37
41	590	0:00:51	271	291	0:00:28	557	1	0:00:00
43	566	0:00:45	277	211	0:00:19	563	183	0:00:16
47	1	0:00:01	281	3	0:00:01	569	329	0:00:25
53	158	0:00:13	283	225	0:00:18			
59	1	0:00:00	293	281	0:00:23			
61	939	0:01:30	331	0	0:00:00			
67	9	0:00:01	337	185	0:00:15			
71	0	0:00:00	347	14	0:00:02			
73	122	0:00:10	349	244	0:00:18			
79	783	0:01:03	353	1	0:00:01			
83	0	0:00:01	359	0	0:00:00			
89	0	0:00:00	367	1	0:00:01			
97	16	0:00:02	397	970	0:01:11			
101	392	0:00:30	401	168	0:00:14			
103	166	0:00:21	409	409	0:00:31			
107	685	0:00:55	419	112	0:00:09			
109	9	0:00:01	421	0	0:00:00			
127	130	0:00:11	431	35	0:00:03			
131	83	0:00:07	433	0	0:00:00			
149	977	0:01:38	439	246	0:00:20			
151	434	0:00:35	443	626	0:00:56			
157	31	0:00:03	449	202	0:00:19			
163	56	0:00:05	457	2	0:00:00			
167	411	0:00:37	461	5	0:00:01			
173	4	0:00:01	463	163	0:00:15			
179	156	0:00:14	467	45	0:00:04			
181	192	0:00:20	473	239	0:00:20			
191	86	0:00:07	479	133	0:00:11			
193	0	0:00:00	487	60	0:00:05			

Table 2 – Solution mutated

Seed	Gen	Time	Seed	Gen	Time	Seed	Gen	Time
1	328	0:00:28	197	336	0:00:26	491	323	0:00:25
2	1	0:00:00	199	536	0:00:40	493	749	0:00:57
3	1001	0:01:15	211	298	0:00:23	497	556	0:00:42
5	196	0:00:15	223	1024	0:01:19	499	395	0:00:29
7	30	0:00:03	227	48	0:00:04	501	291	0:00:22
11	64	0:00:06	229	695	0:00:56	503	356	0:00:32
13	12	0:00:01	233	904	0:01:13	509	553	0:00:42
17	27	0:00:02	239	1232	0:01:35	511	654	0:00:50
19	398	0:00:30	241	169	0:00:12	513	305	0:00:22
23	17	0:00:02	251	630	0:00:50	521	560	0:00:43
29	958	0:01:09	257	239	0:00:20	523	195	0:00:15
31	309	0:00:25	263	253	0:00:19	541	1449	0:01:47
37	130	0:00:10	269	0	0:00:00	547	1367	0:01:43
41	440	0:00:32	271	1548	0:02:11	557	196	0:00:15
43	490	0:00:40	277	285	0:00:24	563	130	0:00:09
47	188	0:00:14	281	309	0:00:24	569	119	0:00:10
53	241	0:00:18	283	215	0:00:17			
59	581	0:00:45	293	418	0:00:36			
61	1022	0:01:21	331	307	0:00:23			
67	714	0:00:55	337	437	0:00:32			
71	69	0:00:06	347	1355	0:01:42			
73	137	0:00:11	349	171	0:00:13			
79	412	0:00:34	353	329	0:00:25			
83	428	0:00:34	359	78	0:00:06			
89	319	0:00:24	367	563	0:00:43			
97	790	0:00:58	397	80	0:00:06			
101	311	0:00:24	401	3346	0:00:32			
103	417	0:00:31	409	126	0:00:10			
107	226	0:00:19	419	273	0:00:22			
109	305	0:00:24	421	66	0:00:05			
127	65	0:00:05	431	444	0:00:34			
131	288	0:00:23	433	72	0:00:05			
149	220	0:00:18	439	1655	0:02:19			
151	37	0:00:03	443	216	0:00:18			
157	1254	0:01:32	449	185	0:00:14			
163	368	0:00:29	457	153	0:00:14			
167	134	0:00:11	461	257	0:00:21			
173	752	0:01:00	463	13	0:00:02			
179	474	0:00:37	467	238	0:00:18			
181	313	0:00:25	473	401	0:00:31			
191	890	0:01:08	479	632	0:00:47			
193	622	0:00:49	487	771	0:00:58			

Appendix 6

This appendix contains PHP implementation of regular expression generator discussed in Chapter 5. The regular expression is built from the training set provided by the user. There are two different generators written in this thesis. First is to convert the jQuery path to its equivalent regular expression pattern and the second is responsible to convert the data format (textual feature).

TS-WIE Automatic Regular Expression Generators (in PHP)

Transformation of the data or path into the equivalent regular expressions is quite complicated. As for the data, it needs to be split into chunk of words, taking into care the white spaces and special characters e.g. dot and brackets. Then these individual regular expression representations have to be combined if the patterns are the same.

It is even more complex with paths transformation. This is because HTML documents structure can be irregular. The weakness of regular expression is to balance the structure, for example, an open tag should have its matching closing tag. Some web page may omit this balancing as it is not mandatory to have it in place e.g. <p> to create a paragraph or having a single tag to indicate an open and close tag e.g.
 tag to insert a line break.

It can also be seen that the resulting regular expressions are not always as simple as *<DIV[^>]*>.*?</DIV>.* The outcome might be an extremely huge regular expression pattern consisting of thousands of alphabets, digits and meta-characters. It is important to note that the regular expression translation is not character by character translation as this will require time in $O(C_1 * C_n)$, where *n* is the total characters. Rather, it is translated word by word except when meta-character exists within the word e.g. 20-21 contains a symbol '-', in which case this will be translated as \d+\-\d+. Meta-character will be handled separately as it requires the escape character '\' to be treated as it should be.

In this approach, the structural (jQuery path) and the textual features of the captured data will be converted to regular expression pattern. Two automatic regular expression generators were developed in this thesis to handle conversion of (i) the structural feature and (ii) the textual feature. Once they are successfully converted, they will be combined together to form one complete regular expression.

The first regular expression generator is to handle the conversion of the structural feature. The steps are:

1. The jQuery path is disjointed (breakpoint is indicated by the separator '>') into tokens.
2. Generate regular expression based on the tokens, applied the following in order (first match wins):
    a. If a token contains 'eq', replace with sets of lowercase open tag, '[^>]*>', closing tag and '.*?' . The number of set required is determined by the integer (index) after 'eq' e.g. tr:eq(1)> shows that the index is 1 and it indicates that the data is in the second

tr. Therefore this requires 2 sets of tr tags. This is translated to *<tr[^>]*>.*?<\tr>< tr[^>]*>*

b. Upon reaching the last tag, '(.*?)' and all balance of the closing tags are added. '(.*?)' indicates the start and stop capture process and it is the text pattern features.

For example, if the path is *div>table>tr:eq(1)>td,* the translation is
<div[^>]><table[^>]*><tr[^>]*>*<\tr>< tr[^>]*> <td[^>]*>(.*)</td></tr></table> *
The above algorithm was implemented in jQuery as below:

```
function convertPath(path) {
   var tags = [];
   var RE_tag_equiv = '';
   var position = 0;
   var lastTag='';
   tags = path.split('>');
   tagsLen = tags.length - 1;
   $.each(tags, function(index, value) {
      if (value.match(/:eq\(\d+\)/g)) {
         value = value.toLowerCase();
         position = value.match(/(\d+)/g);
         value = value.replace(/:eq\(\d+\)/,"");
         var i=0;
         for(i=0; i<=position;i++) {
            if (i != position) {
              RE_tag_equiv += "<"+value+"[^>]*>.*?</"+value+">";
            } else {
              RE_tag_equiv += "<"+value+"[^>]*>";
            }
         }
       } else {
         RE_tag_equiv += "<"+value+"[^>]*>";
       }
       if (index < tagsLen) {
          RE_tag_equiv += ".*?";
       }
       lastTag = value;
   });
   RE_tag_equiv += '(.*?)</' + lastTag +'>';
   return RE_tag_equiv;
}
```

The following describes the conversion of the textual feature into regular expression pattern.

The regex to define the textual format of the extracted information is automatically generated using the following logic (extracted from Conrad (2007) with some modifications on the order and addition of new logic (a, to suit the problem specification in this thesis):

1. Break given example into words (tokens).

2. Generate regex based on the tokens applied in order (first match wins):

If words > 1

a. If it matches pre-defined format, replace with pre-defined format

b. If it's a number, replace with \d+ (1 or more digits)

c. If it's a word with the first letter capitalized, replace with [A-Z][a-z]+ (uppercase letter, followed by one or more lowercase letters).

d. If it's uppercase word, replace with [A-Z]+ (one or more uppercase letters)

e. If it's lowercase word, replace with [a-z]+ (one or more lowercase letters)

f. If it's alphanumeric word, replace with [A-Za-z0-9]+ (one or more alphanumeric letters)

g. If it contains meta-character, split the word into characters and replace with appropriate regular expression notation (alphanumeric or escaped meta-character)

Else: keep the token as a literal regex or an escaped meta-character.

3. Simplify regex (delete regex that match with previous)

The jQuery implementation of textual feature conversion is:

```
function converttoregex(the_text) {
   var RE_data_equiv = "";
   var words = (the_text.split(/(\s+|\(|\)|\!)/)).filter(function(n) {return n.trim()});
   var onechar = the_text.split('');
   var wordcount = 0;
   var counter = 0;
   var the_word = '';
   var pattern = [];

   if (onechar.length > 1) {
     while (counter < words.length) {
       if (words[counter].match(/^(Jan(uary)?|Feb(ruary)?|Mar(ch)?|Apr(il)?|May|
       Jun(e)?|Jul(y)?|Aug(ust)?|Sep(tember)?|Oct(ober)?|Nov(ember)?|
       Dec(ember)?),?$/i)){
         pattern[counter] = (?!Jan(uary)?|Feb(ruary)?|Mar(ch)?|Apr(il)?|May|Jun(e)?|
         Jul(y)?|Aug(ust)?|Sep(tember)?|Oct(ober)?|Nov(ember)?|Dec(ember)?),?";
       } else if (words[counter].match(/\d{1,2}(st|nd|rd|th)$/)) {
         pattern[counter] = "\\d+[(st)(nd)(rd)(th)]\\s+";
       } else if (words[counter].match(/^\d+(,\d{3})*(\.\d{2})?$/)) {
         // match integer or float e.g. 200.20 or 21,000
         pattern[counter] = "\\d+(,\\d{3})*(\.\\d{2})?\\s*";
```

```
      } else if (words[counter].match(/^(\d{1,2})[\/.-](\d{1,2})[\/.-]
        ((\d{4})|(\d{2}))$/)) {
        //match date e.g. 12/12/2013, 12.3.2013, 12-3-2013
        pattern[counter] = "(\\d{1,2})[\/.-](\\d{1,2})[\/.-]((\\d{4})|(\\d{2}))\\s*";
      } else if (words[counter].match(/^((&pound;)|£)\s*( )?\d+/)) {
        pattern[counter] =
          "\\s*((&pound;)|£)\\s*( )?\\d+(,\\d{3})*(\.\\d{2})?\\s*(GBP)?";
      } else if (words[counter].match(/^(\d+)$/)) {
        pattern[counter] = "(\\d+\\s+)+";
      } else if (words[counter].match(/^[A-Z][a-z]+$/)){
        pattern[counter] = "([A-Z][a-z]+\\s*)+";
      } else if (words[counter].match(/^[A-Za-z0-9]+$/)){
        pattern[counter] = "([A-Za-z0-9]+\\s*)+";
      } else if (words[counter].match(/[{}\[\]\)\(^$|*+?\-\\:\.,@]/)) {
        // Break string containing metacharacters into tokens
        // each word is a token, and each symbol (like "@" or "-") is a token
        var char1 = words[counter].split('');
        var charcount = 0;
        var split_pattern = "";
        while (charcount < char1.length) {
          if (char1[charcount].match(/[A-Za-z0-9]/)){
            the_word += char1[charcount];
            if (charcount + 1 == char1.length ||
              !char1[charcount + 1].match(/[A-Za-z0-9]/)) {
                split_pattern += "[A-Za-z0-9]+";
                the_word = '';
             }
           } else {  // Escape all metacharacters - we want to treat as literals
             the_word = '\\' + char1[charcount]+' ?' ;
             split_pattern += the_word;
             the_word = '';
           }
           charcount++;
        }
        pattern[counter] = split_pattern + '\\s*';
      } else {
        pattern[counter] = words[counter] + '\\s*'
      }
      counter++;
    }
  } else { // token is a single character
    if (/\d/.test(onechar)) {
      RE_data_equiv = "\d";
    } else if (/[a-z]/.test(onechar)){
      RE_data_equiv = "[a-z]";
    } else if (/[A-Z]/.test(onechar)){
      RE_data_equiv = "[A-Z]";
    } else if (/[{}\[\]()^\$|*+?-\\:@]/.test(onechar)){
      RE_data_equiv = "\\" + onechar;
    } else {
      RE_data_equiv = onechar;
    }
  }
```

```
   //simplified patterns
   var new_pattern = [];
   var j = 0;
   if (pattern.length != 0) {
       new_pattern[j] = pattern[0];
       j++;
       for(var i = 1; i < pattern.length-1;i++) {
            if (pattern[i] != pattern[i-1]) {
                 new_pattern[j] = pattern[i];
                 j++;
            }
       }
       RE_data_equiv = new_pattern.join("");
     }
     return RE_data_equiv;
}
```

Appendix 7

This appendix contains the algorithm for updating the value of the chromosome. This update is important to make sure that the genome mapped to the same phenome when the rules in the grammar is update due to presence of new token in the example(s) provided by the human user.

Algorithm for Genome Alteration

When there is an update to the list components of a rule, e.g. SECTION tags inserted in the tags rule, the genome, which block mapped to this rule becomes invalid and cannot be used as a seeder to the new initial population. Therefore, this genome needs to be updated. The following algorithm is intended to meet this requirement:

1. Get genome and phenome
2. Find affected block in genome through phenome
3. Determine blockgene = the affected gene
4. Determine mod = the selected rule component
5. newGene = int(gene/numberOfOption) * (numberOfOption + 1) + mod
6. replace gene with newGene
7. repeat step 2 – 6 until all affected blocks are updated

for example :

gene = 9
mod = 1
numberOfOption = 4
therefore the newGene = (9/4) * (4 + 1) + 1
= 2 * 5 + 1
= 11