MULTI AGENT SYSTEM FOR WEB DATABASE PROCESSING

ON DATA EXTRACTION FROM ONLINE SOCIAL NETWORKS

Ruqayya Abdulrahman

Department of Computing

University of Bradford

A thesis submitted for the degree of

Doctor of Philosophy

2012

I hereby declare that this thesis has been genuinely carried out by myself and has not been used in any previous application for a degree. The invaluable participation of others in this thesis has been acknowledged where appropriate.

Ruqayya Abdulrahman

In loving memory of my revered fathers:

Mohammed Saleh and Mohammed Isma'il

(Oh Allah forgive them and gather us in your paradise)

To..

who prays for me days and nights...
my warm-hearted mothers...

who supports me in my long journey...
my lovely husband...

who shares with me my happiness and sadness...
my kind sisters...

who fills my life with joy and hope...
my sweet angels...

Thank you for your belief in me...

Acknowledgements

In the Name of Allah, the Most Gracious, the Most Merciful

All Praise is due to Allah for His Glorious Ability and Great Power for giving me the power, patience and knowledge to complete this doctoral thesis.

In response to the beloved prophet Muhammad PBUH, who said, "One who is not thankful/grateful to people is not grateful to Allah" (Sunan At-Tirmidhi), I would like to express my gratitude to all those who supported and encouraged me to complete this thesis.

First of all, I would like to express my sincere appreciation to my supervisors, Prof. Daniel Neagu and Dr. D.R.W. Holton, for their unlimited guidance, advice and invaluable comments over years. I would not be here without their help. Also, I would like to extend my sincere appreciation to my previous supervisor Prof. Irfan Awan and to the co-authors Mr. Mick Ridley, Dr. Sophia Alim and Dr. Yang Lan for the interesting experience in writing and publishing papers.

My gratitude goes to all my colleagues: Dr. Iman Alansari, Yasmina Bashon, Walid Aladli, Dr. Rasha Osman, Dr. Sahsista, Anna Wojak, Yousef Alqasrawi, Dr. Mona Alkhattabi, Shumaila Alansari, Guzlan Miskeen, Moneera Alnamnakani, Fatimah Alhuzali, Dr. Rasha Fares,

Dr. Amina Alsawai, May Alnashashibi, Manal Aldhohyyan, Nesreen Wali and all friends for their generous time in helping me, listening to me, sharing ideas and supporting me through the course of the thesis.

Also, my grateful acknowledgements are due to Taibah University for the provision of the scholarship, and to the Saudi Cultural Bureau in London for their support.

Last but not least, my deep appreciation to all my family: my mothers; Salma and Zainab, my sisters; Fatimah, Faridah, Kolthoom and Mona, my lovely husband Shehab, my lovely daughters; Aufa, Alaa, Eithar and Albatul, my sisters and brothers in law, and my nieces and nephews who support and pray for me throughout my personal life and career.

Abstract

In recent years, there has been a flood of continuously changing information from a variety of web resources such as web databases, web sites, web services and programs. Online Social Networks (OSNs) represent such a field where huge amounts of information are being posted online over time. Due to the nature of OSNs, which offer a productive source for qualitative and quantitative personal information, researchers from various disciplines contribute to developing methods for extracting data from OSNs. However, there is limited research which addresses extracting data automatically. To the best of the author's knowledge, there is no research which focuses on tracking the real time changes of information retrieved from OSN profiles over time and this motivated the present work.

This thesis presents different approaches for automated Data Extraction (DE) from OSN: crawler, parser, Multi Agent System (MAS) and Application Programming Interface (API). Initially, a parser was implemented as a centralized system to traverse the OSN graph and extract the profile's attributes and list of friends from Myspace, the top OSN at that time, by parsing the Myspace profiles and extracting the relevant tokens from the parsed HTML source files. A Breadth First Search (BFS) algorithm was used to travel across the generated OSN friendship graph in order to select the next profile for parsing. The approach was implemented and tested on two types of friends: top friends and all friends. In case of top friends, 500 seed profiles have been visited; 298 public profiles were parsed to get 2197 top friends profiles and 2747 friendship edges, while in case of all friends, 250 public profiles have been parsed to extract 10,196 friends' profiles and 17,223 friendship edges.

This approach has two main limitations. The system is designed as a centralized system that controlled and retrieved information of each user's profile just once. This means that the extraction process will stop if the system fails to process one of the profiles; either the seed profile (first profile to be crawled) or its friends. To overcome this problem, an Online Social Network Retrieval System (OSNRS) is proposed to decentralize the DE process from OSN through using MAS. The novelty of OSNRS is its ability to monitor profiles continuously over time.

The second challenge is that the parser had to be modified to cope with changes in the profiles' structure. To overcome this problem, the proposed OSNRS is improved through use of an API tool to enable OSNRS agents to obtain the required fields of an OSN profile despite modifications in the representation of the profile's source web pages. The experimental work shows that using API and MAS simplifies and speeds up the process of tracking a profile's history. It also helps security personnel, parents, guardians, social workers and marketers in understanding the dynamic behaviour of OSN users. This thesis proposes solutions for web database processing on data extraction from OSNs by the use of parser and MAS and discusses the limitations and improvements.

Keywords: Data Extraction, Online Social Network, Agent, Multi Agent System, Formal Specification.

Peer Reviewed Publications and Contributions

Journal Publications:

- R. Abdulrahman, D. Neagu, D.R.W. Holton, M. Ridley and Y. Lan (2012), "Data Extraction from Online Social Networks using Application Programming Interface in a Multi Agents System Approach", Transactions on Computational Collective Intelligence Journal, Springer-Verlag (Accepted with Minor Corrections).

- R. Abdulrahman, S. Alim, D. Neagu, D.R.W. Holton and M. Ridley (2012), "Multi Agents System Approach for Vulnerability Analysis of Online Social Network Profiles over Time", Int. J. Knowledge and Web Intelligence, Inderscience (Accepted).

- S. Alim, R. Abdulrahman, D. Neagu and M. Ridley (2011), "Online Social Network Profile Data Extraction for Vulnerability Analysis", Int. J. Internet Technology and Secured Transactions, Vol. 3 No. 2, pp. 194-209, Inderscience. http://dl.acm.org/citation.cfm?id=1972028

Chapters in Books:

- R. Abdulrahman, D. Neagu and D.R.W. Holton (2011), "Multi Agent System for Historical Information Retrieval from Online Social Networks", Agent and Multi-Agent Systems: Technologies and Applications, J. Oshea K. Crockett R. Howlett J. Lakhmi (ed.), ISBN: 978-3-642-21999-3, pp. 54-63, Springer, Berlin.
  http://www.springerlink.com/content/b74j67022w348161/

- R. Abdulrahman, D. Neagu, D.R.W. Holton and M. Ridley (2011), "Formal Specification of Multi Agent System for Historical Information Retrieval from

Online Social Networks", Agent and Multi-Agent Systems: Technologies and Applications, J. Oshea K. Crockett R. Howlett J. Lakhmi (ed.), ISBN: 978-3-642-21999-3, pp. 84- 93, Springer, Berlin. http://www.springerlink.com/content/t3182685257503v2/

Conference Papers:

- R. Abdulrahman, S. Alim, D. Neagu and M. Ridley (2010), "Algorithms for Data Retrieval from Online Social Network Graphs", Proceedings of the 10th International IEEE Conference on Computer and Information Technology (CIT 2010), 29th June - 1st July 2010, Bradford, UK, pp.1660-1666, IEEE CS.

- S. Alim, R. Abdulrahman, D. Neagu and M. Ridley (2009), "Data Retrieval from Online Social Networking Profiles for Social Engineering Applications", Proceedings of the 4th International Conference for Internet Technology and Secured Transactions (ICITST 2009), 9th - 12th November 2009, London, UK, pp. 207- 211, IEEE Xplore.

Conference Contributions:

- R. Abdulrahman, D. Neagu and D.R.W. Holton (2011), "A Better Way to Retrieve Information from Online Social Networks", Proceedings of the 5th Saudi International Conference, 23rd -26th June 2011, Warwick, UK. (**Awarded the Prize of Second Best Poster**).

- R. Abdulrahman, and D. Neagu (2010), "An Investigation in using Multi Agent Systems for Data Retrieval from Online Social Network ", Proceedings of the 4th Saudi International Conference, 29th - 31st July 2010, Manchester, UK.

Presentations:

- Data Extraction from Online Social Networks using Application Programming Interface in a Multi Agents System Approach (2011) University of Bradford, UK

- The Institution of Engineering and Technology (IET) prestige invited talk: Algorithms for Social Engineering in Online Social Network (with S. Alim, D. Neagu and M. Ridley), (2010) University of Bradford, UK. http://www.theiet.org/local/uk/yorks/west/social-eng.cfm

- Open Day: Automated Data Retrieval from Online Social Network Profiles (with S. Alim), (2010) University of Bradford, UK.

- Presentation at FAIRS2009 for: the 3rd annual forum for AI research students: Algorithm for Data Retrieval from Online Social Network Profiles (2009) Cambridge University, UK.

- Presentation and Demonstration to students from Bradford Grammar School: Data Extraction from Online Social Network Profiles (2009) University of Bradford, UK.

- AI Research Seminar: Data Retrieval from Online Social Networking Profiles for Social Engineering Applications (2009) University of Bradford, UK.

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
ASP	Active Server Pages
BFS	Breadth First Search
DE	Data Extraction
DR	Data Retrieval
DOM	Document Object Model
gAg	grabber Agent
gmAg	group manager Agent
GUI	Graphical User Interface
HTML	HyperText Markup Language
IE	Information Extraction
IR	Information Retrieval
JSON	Java Script Object Notation
mAg	master Agent
MAS	Multi Agent System
OSN	Online Social Network
OSNRS	Online Social Network Retrieval System
PHP	PHP: Hypertext Preprocessor
PIW	Publicly Indexable Web
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language
URL	Uniform Resource Locator
WWW	World Wide Web

Contents

List of Figures

List of Tables

Chapter 1

Thesis Introduction

1.1 Introduction

When the World Wide Web (WWW) or “web” for short was released to the public in 1993, the early web stage which is represented by “web 1.0” was started. It was considered as a “read-only web”, according to Tim Berners-Lee, due to the fact that its web pages usually contained text and hypertext: text with hyperlinks to point to other text or documents.

The lack of active interaction between users and the web led to the birth of “Web 2.0” in 2004 and to the beginning of the “read-write web” allowing users to contribute with the information available to them. Online Social Networks (OSN) such as Facebook[1], Twitter[2] and Myspace[3] are some of the most commonly referenced phenomena to explain the concept of Web 2.0, where users are encouraged to generate web content in the form of videos, photographs and text posted as comments and tags.

Consequently, this has led to a massive explosion in the amount of data available on the web and challenges researchers to apply new methods to reverse the process in order to gain the benefit of these data. Many theories have been applied to produce

[1] http://www.facebook.com/
[2] http://www.twitter.com/
[3] http://www.myspace.com/

services and tools for end users to allow them to manage, query and extract data from the web. This thesis plays a part in this, with its particular contributions to extracting data from OSN profiles through investigating the current techniques and developing new approaches of extraction. The approaches developed are general in that they could be applied to any OSN. Furthermore, they are not restricted to OSNs sites, but to many different area such as what is described in Section 3.2.1.

The purpose of this chapter is to present an overview of this research study. The problem domain is presented in Section 1.2. The motivation, aims and objectives of the research are given in Sections 1.3 and 1.4 respectively. Section 1.5 explains the research methodology. Finally, the structure of the thesis is outlined in Section 1.6.

1.2 Problem Domain

Web data extraction is a field which concerns the extraction of data from different web resources including websites, online databases and services. With the explosion in the amount of data made available online, end users and application programs face difficulties in finding useful and relevant data. This issue encourages researchers to develop new methods for extracting data from the web.

OSN is one of the fields where the amount of personal data which is publicly available online has increased massively. This is due to OSN allowing millions of individuals to organize, find friends and share their personal information and interests. This information can provide a good collection of study resources for researchers from different disciplines such as psychology, sociology and computer science.

There are several issues which contribute to users and applications failing to find the required data. One of these issues is related to the representation of information. Data on web pages can be found in different formats. HTML is designed for unstructured data which contains information in several formats, e.g. text, image, video and audio. It is known that web pages in HTML format are "dirty" due to the contents being ill-formed and "broken" [95]. In contrast, XML and XHTML are designed for more structured data. They are stricter in terms of having well-formed documents: i.e., the documents' contents should conform to their syntax rules. This feature helps the parsers of search engines to interact with the web pages' contents more efficiently [95].

Another issue which prevents efficient data extraction is related to the technique used by search engines to find related web pages. Search engines depend on crawlers to search the WWW for the required keyword(s) entered by end users. The majority of the web pages which contain useful data are in the Hidden Web or Deep Web, which is outside what is called the Publicly Indexable Web (PIW) which is reachable by search engine crawlers.

1.3 Motivation

Developing new methods for extracting data from the web in general and OSN in particular has attracted many researchers for years. What makes OSN profile data different is that its structure, presentation and contents change rapidly and continuously over time. Moreover, the new facilities to access these OSNs through mobile devices, smartphones etc., especially when the developers pay great attention to making access to OSN sites one of the leading features and applications of these devices, add extra challenges for researchers in contributing to the development of new methodologies to extract and monitor these changes over time.

Most research in the field of OSN data extraction either extracted data manually through surveys and interviews or did not mention how the data was collected. There is little research associated with automated extraction methods from OSN.

Moreover, although there are studies which attempt to extract millions of profiles from different OSN, to date, they have analyzed the results of data collected by visiting each profile only once. To the best of the author's knowledge, there is no reported work which deals with the monitoring of changes in OSN profiles over time automatically.

In contrast, the Multi Agent System (MAS) has many features such as autonomy, sociability, pro-activity and perceptivity, which have been used to solve problems related to real-time and distributed systems perfectly. These features fit very well with the problem stated above in offering the possibility to distribute agents over the OSN profiles to extract data and monitor any updates in the profile.

1.4 Aims and Objectives

This thesis aims to investigate new approaches in automated data extraction from OSN, then save the results in a local repository to be used for further offline analysis. In addition, the thesis aims to investigate the ability to apply MAS in a distributed environment such as an OSN server. In order to achieve these aims, the objectives of the thesis are set as follows:

- Provide an algorithm to extract semi-structured and unstructured data from an OSN source web page.
- Provide an algorithm to extract data from an OSN in the absence of the source of the web page.
- Develop a decentralized approach using MAS for automated extraction of data from an OSN.
- Develop an approach to effectively monitor updates in OSN profiles over time.

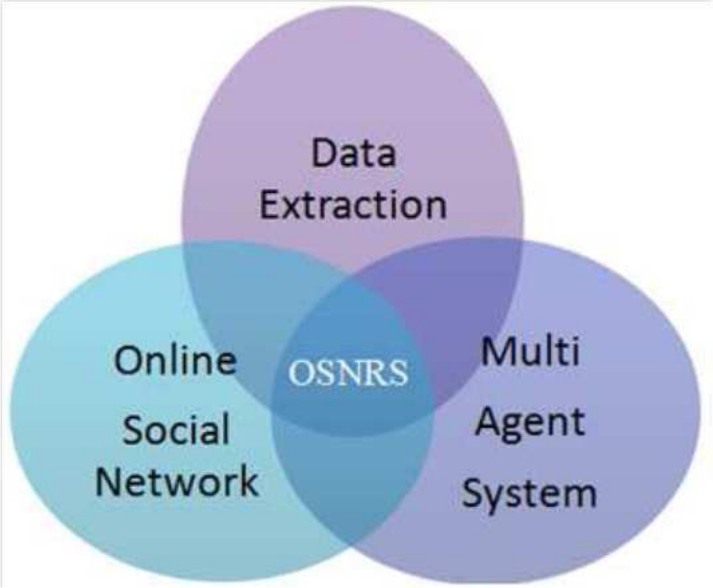


Figure 1.1: Venn Diagram of Intersection between OSNRS Areas

- Create historical records of the OSN profile records to help researchers of various disciplines such as security personnel, parents, guardians, social workers and marketers in understanding the dynamic behaviour of OSN users.
- Study the accuracy of extracted data.

1.5 Research Methodology

Based on the given aims and objectives, the research methodology of the thesis will be to initially gain a brief understanding of the following disciplines as shown in Figure 1.1:

- What: the process of extracting data.
- Where from: OSN, one of the most recent, popular and dynamic web resources.
- How: using MAS technique combined with the parser or API.

Consequently, a review of the state of the art within these topics and the intersection between them is presented to motivate the need for a system which can extract historical data from OSN profiles and monitor changes in these profiles over time. In order to address this goal, the thesis research was structured into the following four phases:

Phase 1: Implementing an application to extract semi-structured and unstructured data from OSN source web pages using the parser.

Phase 2: Presenting the feasibility of using MAS technology in extracting historical data from OSN sites through using formal specification.

Phase 3: Enhancing the application developed in Phase 1 by applying MAS to implement a decentralized application as specified in phase 2.

Phase 4: Improving the application developed in phase 3 by replacing the parser with API in order to be able to extract data from the OSN in case of absence of the source code.

This study will serve as a base for future studies in the process of tracking user profile history and understanding the behaviour of OSN users, especially when combined with text mining.

1.6 Thesis Organization

The thesis has eight chapters as shown in the Figure 1.2. They are as follows:

Chapter 1: provides a context for the research area. It presents a brief introduction of the problem domain and overview of the research approach and methodology followed by the layout and content of the thesis.

Chapter 2: presents a background of the three main disciplines and the intersection of which this thesis lies; web data representation and extraction, Online Social

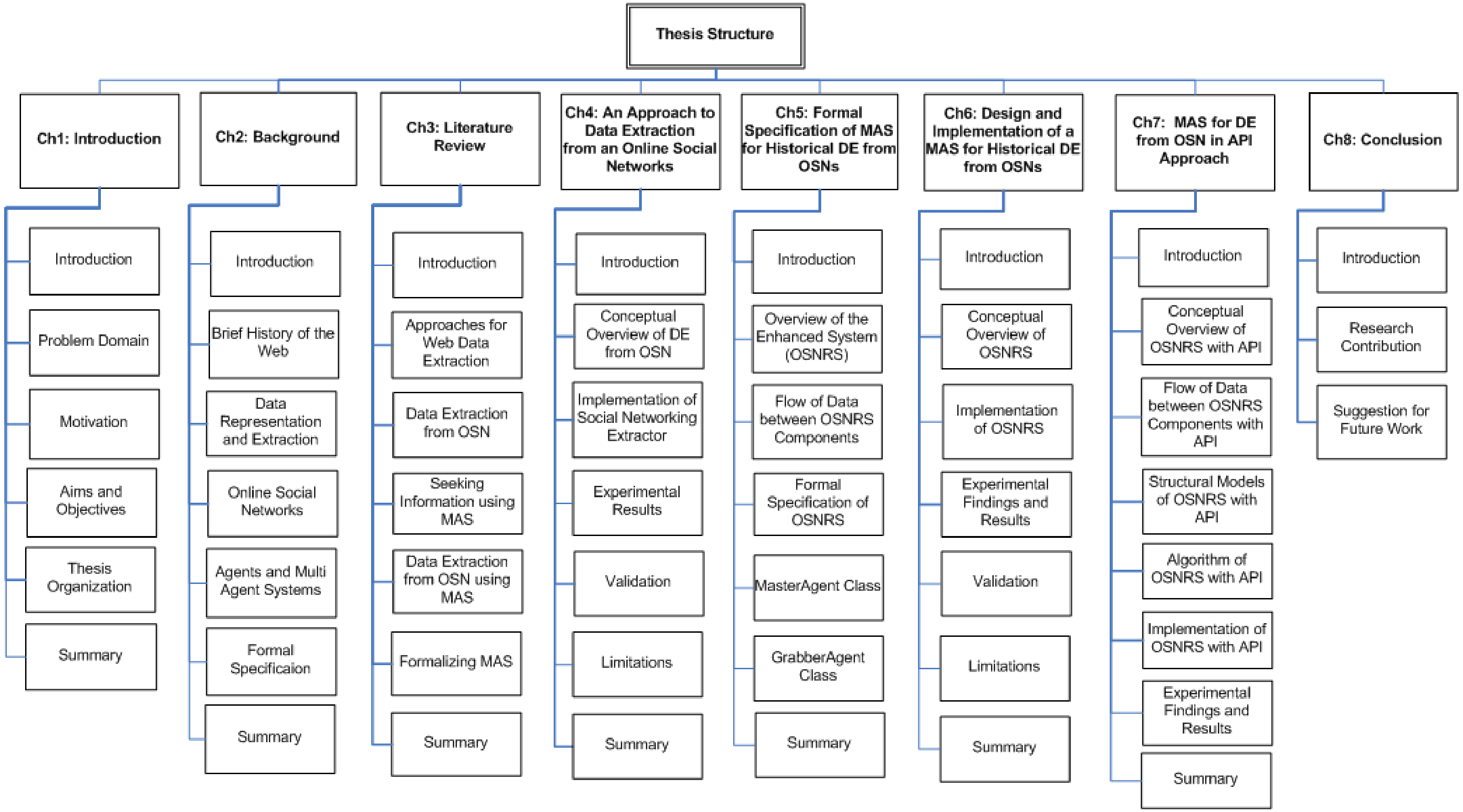

Thesis Structure
Ch1: Introduction
Introduction
Problem Domain
Motivation
Aims and Objectives
Thesis Organization
Summary
Ch2: Background
Introduction
Brief History of the Web
Data Representation and Extraction
Online Social Networks
Agents and Multi Agent Systems
Formal Specificaion
Summary
Ch3: Literature Review
Introduction
Approaches for Web Data Extraction
Data Extraction from OSN
Seeking Information using MAS
Data Extraction from OSN using MAS
Formalizing MAS
Summary
Ch4: An Approach to Data Extraction from an Online Social Networks
Introduction
Conceptual Overview of DE from OSN
Implementation of Social Networking Extractor
Experimental Results
Validation
Limitations
Summary
Ch5: Formal Specification of MAS for Historical DE from OSNs
Introduction
Overview of the Enhanced System (OSNRS)
Flow of Data between OSNRS Components
Formal Specification of OSNRS
MasterAgent Class
GrabberAgent Class
Summary
Ch6: Design and Implementation of a MAS for Historical DE from OSNs
Introduction
Conceptual Overview of OSNRS
Implementation of OSNRS
Experimental Findings and Results
Validation
Limitations
Summary
Ch7: MAS for DE from OSN in API Approach
Introduction
Conceptual Overview of OSNRS with API
Flow of Data between OSNRS Components with API
Structural Models of OSNRS with API
Algorithm of OSNRS with API
Implementation of OSNRS with API
Experimental Findings and Results
Summary
Ch8: Conclusion
Introduction
Research Contribution
Suggestion for Future Work

Figure 1.2: Thesis Structure

Network and Multi Agent System. Also a brief overview of formal specification is presented.

Chapter 3: reviews the literature and the state of the arts: web data extraction, OSN and MAS.

Chapter 4: describes the thesis first contribution towards extracting data from the web. In particular, it presents the first algorithm to extract data from OSNs automatically. The conceptual overview and experimental work of the developed approach is described in detail, accompanied by the results. The contributions from this chapter are published in: [20, 9, 19] and presented in:

- The Institution of Engineering and Technology (IET) prestige invited talk: *Algorithms for Social Engineering in Online Social Network* (with S. Alim, D. Neagu and M. Ridley), (2010) University of Bradford, UK. `http://www.theiet.org/local/uk/yorks/west/social-eng.cfm`
- Open Day: *Automated Data Retrieval from Online Social Network Profiles* (with S. Alim), (2010) University of Bradford, UK.
- Presentation at FAIRS2009 for: the 3rd annual forum for AI research students: *Algorithm for Data Retrieval from Online Social Network Profiles* (2009) Cambridge University, UK.
- Presentation and Demonstration to students from Bradford Grammar School: *Data Extraction from Online Social Network Profiles* (2009) University of Bradford, UK.
- AI Research Seminar: *Data Retrieval from Online Social Networking Profiles for Social Engineering Applications* (2009) University of Bradford, UK.

Chapter 5: highlights through formal specification (Object-Z), the feasibility of using MAS technology in extracting data from OSN sites in order to ensure that the proposed web based system, the Online Social Network Retrieval System (OSNRS), is robust, reliable and fits its purpose, before implementing the application. The contribution from this chapter is published in [10].

Chapter 6: improves the algorithm presented in Chapter 4 through applying MAS technology in extracting data from OSN. An algorithm to implement the approach developed for the OSNRS is proposed. The conceptual overview and experimental work of the developed approach is described in detail, accompanied by findings and results. The contributions from this chapter are published in [8, 13, 11].

Chapter 7 continues the previous work of OSNRS, which provides real-time monitoring of OSN profiles, through proposing new algorithms using API in order to overcome the limitations of parsers. This facility allows OSNRS agents to attain the required attributes despite modifications in the representation of the profiles' source web pages. The experimental work applies to a Facebook mock network as well as a real network. The contribution from this chapter is published in [14, 12] and presented in:

- Data Extraction from Online Social Networks using Application Programming Interface in a Multi Agent System Approach (2011), Student Seminar, University of Bradford, UK

Chapter 8 presents conclusions regarding the contribution of the thesis and discusses the limitations of the research. Finally, future areas of work are suggested.

1.6.1 Ethical Issues and Data Extraction form OSN

Associating ethics with DE from OSN is one of the main concerns of this thesis due to personal data is being accessed directly. The study can be divided into two stages regarding the OSN targeted in the experimental work for this thesis. Firstly, when data is extracted from Myspace OSN, only public profiles have been crawled and parsed. Private profiles were never been accessed except what is provided by Myspace as will be explained in Section 4.5.

Secondly, in case of Facebook OSN which requires user authorization and authentication to extract data, the author filled a “Research Ethics Application Form” to gain approval from “The Committee for Ethics in Research” at the University of Bradford. Appendix (A) illustrates the Consent Form which informs the participants about the proposed research.

NOTE: All clear screenshots of OSN real profiles contained in this thesis is presented temporary in order to illustrate the idea to the examiners. These figures will be blurred in the final submission of the thesis to avoid breaching privacy.

Chapter 2

Background

2.1 Introduction

The aim of this chapter is to introduce the fundamental concepts of the three main areas for the thesis; Data Extraction (DE), OSN and MASs which are shown in Figure 1.1. However, it is firstly necessary to highlight relevant topics which are important in the explanation of the three areas.

Thus, Section 2.2 presents a brief history of the web, while data representation and extraction from the web is demonstrated in Section 2.3. The context of the extraction which is OSN, is detailed in Section 2.4. Section 2.5 describes agents and MASs as an approach for DE. The formal specification concept is presented in Section 2.6 in order to construct a mathematical model of the proposed system. Finally, the summary of the chapter is presented in 2.7.

2.2 Brief History of the Web

In the last two decades, we have witnessed the beginning of the explosion in the digital information with the rise of the web [91](p. 1). Most people use the words “web” and “Internet” interchangeably as synonyms while they are not. The Internet is a networking protocol which connects computers over the world through a physical network, while the web is a software protocol that runs over the Internet to build a virtual network linking a massive amount of information stored in files together. Briefly, the web has a memory which stores a network of interwoven documents through links [121, 73, 57](p. 1-3, p. 5, p. 311 respectively).

However, the original idea of the web, which is the general domain of this thesis, could be traced back to 1945, when Vannevar Bush described his imaginary desktop machine which is called a "memex" to store information in a local file system. The items on the machine are linked to each other repeatedly to form a trail which allows memex users to traverse it for sharing and exchanging knowledge through "associative indexing". The idea of a memex has been improved by Bush and other researchers until Ted Nelson generalized the memex system after 20 years to coin the term "hypertext" and built his system "Xanadu". Nelson viewed his system as a network of unlimited size of repositories that allow users to connect their documents to any document on the network and achieve a universal hypertext [91](p. 3-4).

This view came true three years later in 1968 when the first online hypertext system was developed by Douglas Engelbart. Then, it turned into reality as known today by Tim Berners-Lee who invented the WWW as "*information space where data of all types could be freely accessed*". Bernars-Lee introduced his first web browser at the end of 1990, including a server and an editor to write the webpage, and then Marc Andreessen developed "Mosaic" in 1993 as a free web browser accessible by the public [91](p. 3-4).

The webpage usually contains text and the hypertext (text with hyperlinks which are able to point to another document or any part of the document). Individual web pages define the beginning of the web forming what are known as websites. Since most websites require updating and feeding with new information continuously, there is a need to handle, organize and access this information within websites.

The database concept has proved to solve such problems of managing and querying information on local machines and over networks. Thus, researchers were attracted to gather the power of the web and database technology in order to produce what is called a web database [57](p. 31). Web databases could be described as a database whose data is managed and accessed through the Internet. In other words, "accessible online database" as defined by Goa et al. in [62]. Many HTML pages are a visible web database. Web database applications have been developed to enable data to be managed and present analytical results online. However, this data need to be extracted in the first place. Web data extraction is a reverse engineering process of how this data is built, structured and represented on the web as will be detailed in the following section. However, since some research in this field distinguishes between data and information, and extraction and retrieval, it is important to clarify the relationship between these terms.

Clarifying Terminologies

As shown in Figure 2.1, each area of these 4 areas is made up from a combination of two broader areas; Data vs. Information and Extraction vs. Retrieval. The following sections distinguish each of them briefly.

1. Data Extraction (DE)

Data Extraction is also called "Web Scraping". Gao et. al. in [62] stated that web data extraction is extracting "*the relevant information from a variety of different Websites belonging to identical fields, and then integrat(ing) them into a unified format for the following-up treatment or application*". DE is concerned about retrieving data out of unstructured or poorly structured source to enable further processing or storage. Unstructured data may include documents, emails, scanned text and web pages.

The main technical challenge that affects DE from unstructured data is the need to deal with the continuous changes in the physical formats of the page. However,

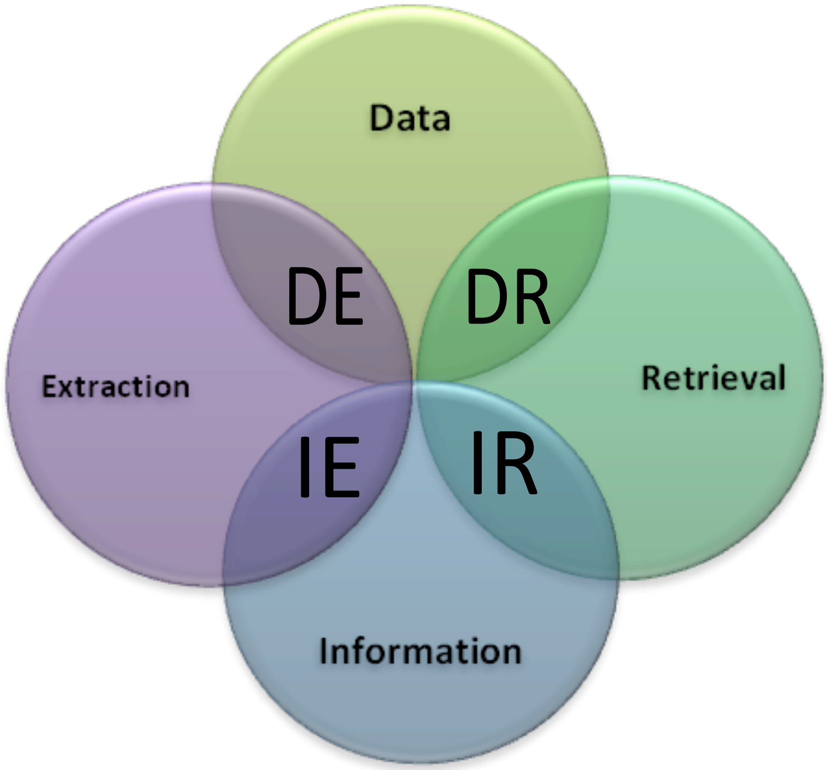


Figure 2.1: Clarifying Terminologies of Information Seeking

most webpage designers adding structure to unstructured data in different forms such as:

- Using regular expression in records to identify small or large scale structure.
- Using table-based approach to categorize common sections in a limited domain.
- Using text analytics to help in understanding the text.

Automated data extraction concerns extracting information which is relevant to a user in a concise form.[17].

2. Data Retrieval (DR)

It could refer to different processes such as recovery of lost data or gathering information from unknown individual and organization of data. Mostly, DR addresses extracting the required data from Databases, such as reports and queries, then storing the output in a file or printing them on the screen.

3. Information Retrieval (IR)

IR focuses on finding and ranking a subset of documents from a given collection of documents based on the user's query. Consequently, the user has to browse the returned documents to fulfil his needs [113]. The information could be in documents, meta data about documents or even the documents themselves which are found in a rational database or other WWW. The most well known application of IR is in search engines which will be explained in the coming sections.

There are overlaps between Data Retrieval, Document Retrieval, Information Retrieval and Text Retrieval but they differ [17].

4. Information Extraction (IE)

IE means "*the identification and extraction of instances of a particular class of events or relationships in a natural language text and their transformation into a structured representation e.g. database*" [113]. In other words, IE aims to extract structured information from unstructured machine readable documents such as news wire reports. IE and IR complement each other and borrow techniques from each other [113].

According to these classifications, it would be clear that this thesis focuses on the DE although DR could be used as well. This is due to in early stages the target of web pages for this thesis vary between unstructured and semi-structured formats which are not applicable for DR. In contrast, since the data retrieved is stored basically on web database, it is possible to use DR. However, the terms data and information may be used interchangeably in the thesis regarding their general meaning.

2.3 Data Representation and Extraction

There are several issues which contribute to users and applications failing to find the required web pages for the information they are seeking. One of these issues is related to data representation and the other issue is related to the technique used to find related web pages which contain the required information. Each of these issues will be discussed separately in the following sub sections.

2.3.1 Data Representation on the Web

Web pages can be categorized into either static or dynamic. Briefly, static means the web page contents are displayed as they are stored in ordinary documents. In contrast, a dynamic web page is a web page whose contents are generated at the time of request by web applications. Table 2.1 summarizes some of the features of static and dynamic web pages as well as a comparison between these two categories [117, 85].

The progression of moving from static web pages to dynamic web pages comes along with the changes in the format of the web pages. Most web pages are written in HyperText Markup Language (HTML). The typical HTML, as shown in Figure 2.2(A), relies on a set of markup tags such as <h1> and <img> in order to describe web pages. HTML is designed for unstructured data, which is ill-formed and broken [95].

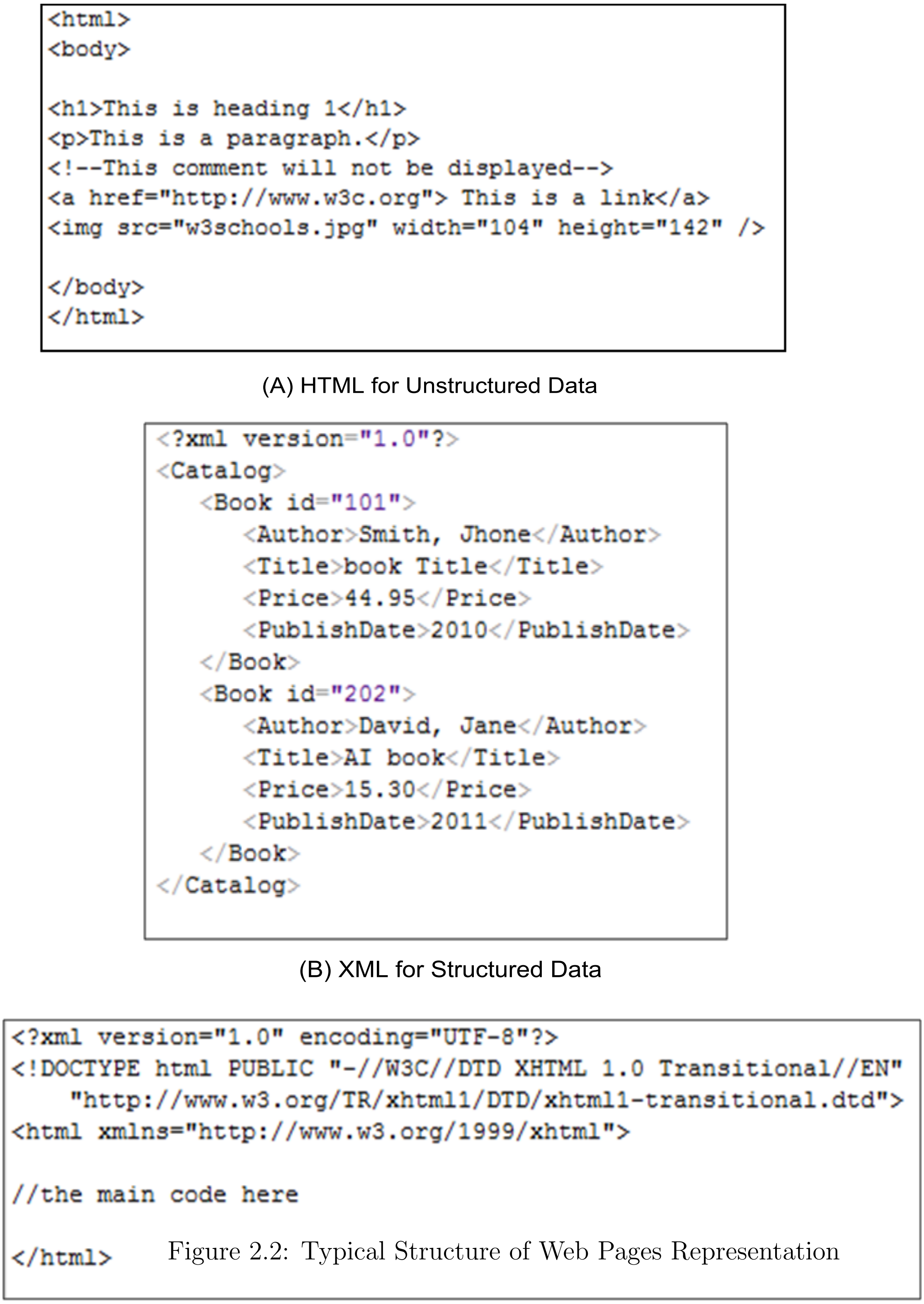
However, from the beginning of last decade , there was a move to be stricter regarding web page design as W3C[1] recommended web designers to use the eXtensible Markup Language (XML) and eXtensible HyperText Markup Language (XHTML) as shown in Figure 2.2(B) and 2.2(C) respectively [2].

XML and XHTML are designed for more structured data, i.e. structured and

[1] http://www.w3.org/

	Static web pages	**Dynamic web pages**
Features	• Contents are written as ordinary files. • Contents not changing. • Contents and presentation are not separated. • Some page structure is replicated. • Difficult to update contents. • Mostly have extensions “.php” or “.asp” or the URL contains a query string.	• Contents provided by scripting features (php, Microsoft ASP). • Contents and presentation are separated. • Contents and presentation may be unique every time it is requested. • Contents may often update as a result of database query. • Mostly have extensions “.htm” or “.html”.
Advantages	• Easy to create and manage through using standard operating system tools and editors. • Fast getting the contents from a file to a network. • Accessible by most traditional search engine robots for indexing.	• All sites’ structure and visual design could be controlled via creating single or few templates. • Easy to maintain structure and contents.
Dis-Advantages	• Content providers should create webpage from scratch. • Web site structure should be maintained manually or using tools e.g. “Microsoft FrontPage”. • The tools are complex to learn and may be expensive for creating personal websites.	• Only major search engine robots can index the web page. • Local indexing and searching are harder.

Table 2.1: Comparison between Static and Dynamic Web Pages



```
<html>
<body>

<h1>This is heading 1</h1>
<p>This is a paragraph.</p>
<!--This comment will not be displayed-->
<a href="http://www.w3c.org"> This is a link</a>
<img src="w3schools.jpg" width="104" height="142" />

</body>
</html>
```

(A) HTML for Unstructured Data

```
<?xml version="1.0"?>
<Catalog>
   <Book id="101">
      <Author>Smith, Jhone</Author>
      <Title>book Title</Title>
      <Price>44.95</Price>
      <PublishDate>2010</PublishDate>
   </Book>
   <Book id="202">
      <Author>David, Jane</Author>
      <Title>AI book</Title>
      <Price>15.30</Price>
      <PublishDate>2011</PublishDate>
   </Book>
</Catalog>
```

(B) XML for Structured Data

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

//the main code here

</html>
```

Figure 2.2: Typical Structure of Web Pages Representation

(C) XHTML for Semi Structured Data

semi-structured data. They are stricter in terms of having well-formed documents i.e., the documents' contents should conform to their syntax rules. For example, elements must be: properly nested, always be closed, must be in lowercase and its documents must have one root element. This feature helps the parsers of search engines to interact with the web pages' contents more efficiently [95, 2]. More details will be discussed in Chapter 4.

2.3.2 Seeking Information on the Web

In contrast to the popular view of information seeking on the web as being just a few mouse clicks, information seeking from the web is different and more difficult than it from other electronic systems such as libraries or databases. This is due to the nature of the web itself and its features e.g. the size of the web, the dynamic and heterogeneity nature of its contents, the organization and representation of information within web pages, in addition to the rapid changes of all these features. [99](p. 7).

Initially, the broad term "information seeking" is used as Levene suggests in [91](p. 24) in order to refer to the process of finding information on the web. Seeking information could be traced back to the period between late 50s and early 60s. On those days, there was a thought that the traditional techniques of searching would not be successful on unstructured and open environments like the web. Thus, early search tools missed most of the basic capabilities for seeking information [122](p. 3).

There are two main strategies for seeking the required information on the web as Figure 2.3 shows; navigating and searching. A brief overview of each strategy is presented below.

1. Navigating

Navigating the web is known also as browsing or surfing the web. Navigating means "*employing the link-following strategy starting from a given web page, to satisfy the information need*" [91](p. 24-25). The link points to another object on the web such as an image, document, webpage, etc. The navigating is either directly or through using a directory.

Direct Navigation: is the simplest and often most successful approach to find the home pages of companies, institutes and services but rarely successful for finding products. It depends on the user typing the website address in the browser, then following the links to find the required information depending on the (proximal) cues like text and images snippets that the web pages are designed to offer to their users. Some browsers share auto-complete feature to help users when they are typing the websites URLs.

Navigation within a Directory: a web directory could be considered as like a book's table of contents. They are "*catalogues of information resources that are*

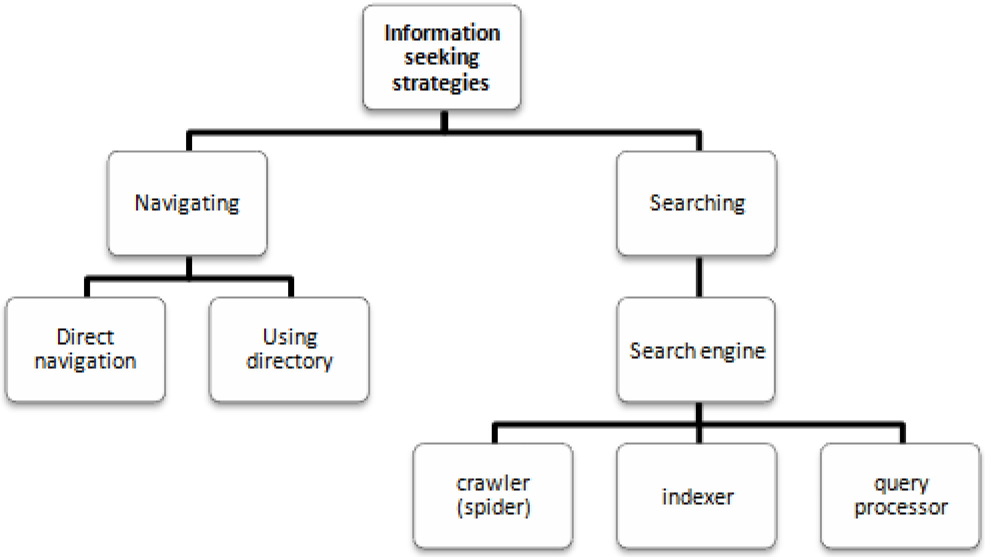


Figure 2.3: Strategies of Information Seeking

arranged by subject, which makes them very useful for browsing" [73](p. 54). This approach of navigation is the basis of the "Web Portal" sites which work as a gateway to other resource on the web. It depends on human editors to organize the information in the sites and classify them in directories [91], [122](p. 22).

2. Searching

This strategy is the most common in use nowadays if not the only one used by most people. It is based on getting the required information by employing a search engine. Sherman and Price in [122](p. 23) defined search engines as "*databases containing full-text indexes of web pages*". Search engines work through three parts as follow:

1. **The web crawler:** which finds and fetches the web pages. (More details in next sections).

2. **The indexer:** which indexes the web pages' words and stores them in massive databases.

3. **The query processor:** which recommends the documents based on the best match of the submitted query and the stored indices in the database [122] (p. 26).

The common steps for using a search engine may include:

1. Submit the query which consists of keyword(s) to the search engine.
2. Select one of the results that are ranked and referenced by search engines.
3. Surfing the web pages by clicking from summary shown links related to the query .
4. Interrupt navigation and modify the query to be more specific based on the personal judgement of the user when surfing the result.
5. Go back to first step.

Navigating	Searching
• Easy, natural, intuitive. • Impractical with big size, diversity of web. • Poor for exhaustive searches.	• Requires learning how to use tools to get satisfactory results. • No limitation in the size of the web. • Good for exhaustive searches.

Table 2.2: Comparison between Information Seeking Strategies

Table 2.2 presents a comparison between the strategies. Nevertheless navigating or searching, the process of seeking information faces some problems such as:

- The web is an open, unstable system. i.e. the format and the contents of web pages and web sites change continuously.
- The quality of the presented information varies. No controls on who is publishing, how accurate is the published information, are the links still available or they are out of date.

To overcome such problems, researchers keeping developing techniques to simplify the process of seeking information as this thesis aims. Thus, the following sections present some of the most common techniques which help in the process of DE from the web.

2.3.3 Web Data Extraction Techniques

As stated previously, the fast growth and the decentralized nature of the web brings major problems in term of navigation and making use of indexing the full text of pages that are stored on web servers [122](p. 12-13). In Section 2.3, it has been mentioned that the second issue which prevents efficient data extraction is related to the technique used to find the related web pages which contain the required information. The following sections present some of the techniques (crawlers, wrappers

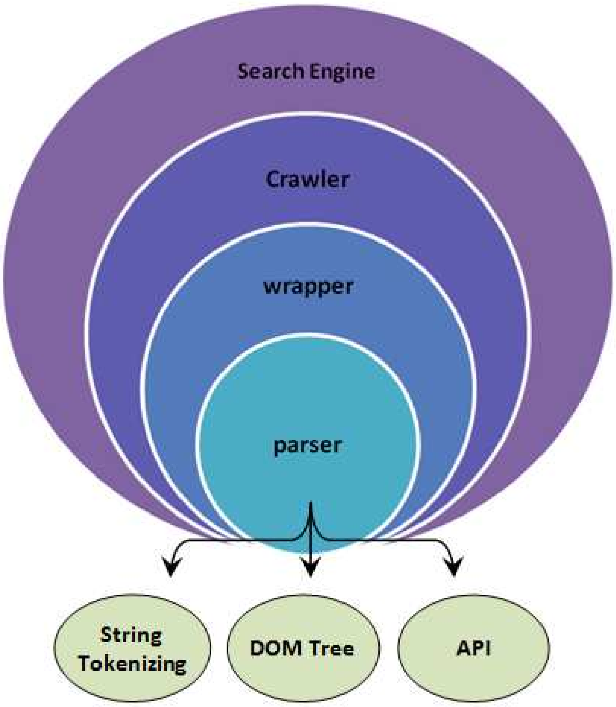


Figure 2.4: Relationship between Different Techniques Graph 2

and parsers) as shown in Figure 2.4. Note that the relationship between these techniques is a containment relation from the outer to the inner. This means that each technique includes the next one in order to extract data from various formats of documents on the web. In addition, the parser is based on either the string tokenizing, DOM or API. DOM is explained first because it is the most used with these techniques.

Document Object Models (DOM)

The Document Object Model is a W3C standard for defining (and accessing) the objects, properties and methods of all the elements in a document. The professional definition of DOM is a "*platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page*"[75].

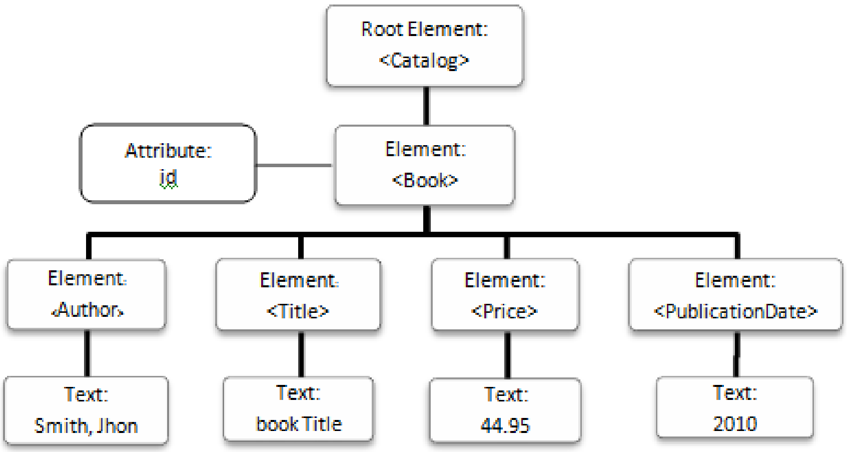


Figure 2.5: The DOM Node Tree Corresponding to XML File in Figure 2.2(B) Modified from [3]

A DOM approach could be used to parse 3 different types of documents. HTML documents, XML documents and any structured documents. DOM views the document as a node tree where each element in the document represents a node and has a relationship with other elements (nodes)[1].

Figure 2.5 shows the DOM node tree corresponding to XML documents in Figure 2.2(B), modified from [3].

1. Crawlers

Search engines depend on crawlers (also known as spiders or bots) to search the WWW for the required keyword(s) which are entered by end users. The crawler is "*a computer program that navigates the hypertext structure of the web*" [17](p. 263). Crawlers collect information about the visited web pages then save them in a queue. This information is recorded in a process called indexing which is used later for ranking websites. Crawlers will be activated periodically to update the index. The basic algorithm of a crawler is as follows:

1. Get the first seed URL from the list of URLs to be visited.

2. Fetch the web page of the seed URL.

3. Index the web page.

4. Add new URLs found on the web page to the list of URLs.

5. Go back to first step.

During this process, crawlers have to address problems such as detecting duplicate web pages which have different URLs, determining the quality of the fetched web page and indexing web pages which contain errors. Even though, there is a problem that has arisen because of the limitation of crawlers' capabilities. Crawlers can cover only the publicly indexable web (PIW) while the majority of useful data is in the hidden or deep web, which is not reachable by crawlers as highlighted by Lawrence and Giles in [89], and Bergman in [31]. Deep web pages are described by Park and Barbosa in [111] as dynamic pages listing data from databases using a predefined format. Their content is likely to be of very high quality since they are managed by organisations interested in maintaining accurate and useful databases.

Due to the fast and continuous growth in the size of the web, it is recommended to use the focused crawlers which focus on visiting web pages related to topics of interest based on the highest ranking URLs in the list to crawl first [91].

2. Parsers

Ludwig in [94] described the parser as a tool that "*breaks data into smaller elements, according to a set of rules that describe its structure*". In other words, it takes the HTML source of webpage as a stream of Unicode characters to a tokenization stage where the tokens are classified to build the DOM tree [3].

Gupta et al. mention several advantages of parsing a webpage's HTML into a DOM tree such as the ability to extract data from large logical units, manipulate smaller units such as specific links within the structure of the DOM tree and reconstruct a complete webpage easily.[67].

3. Wrappers

A wrapper is considered to be a traditional and useful approach for DE. It based on converting HTML documents into semantically meaningful XML files to simplify the operation of extracting data [87, 50, 92]. Formally, a wrapper is "*a function from a page to the set of tuples it contains*"[86].

To create a wrapper, we have to specify the structure of the source webpage through:

- Identifying the interesting tokens (e.g. heading of sections) on the page.
- Identifying the nested hierarchy within the sections and/or subsections of the source webpage.

The resulting structure will be a basis on which to build a parser which is described in the previous section [24].

However, using wrappers is impractical in some cases where the interesting tokens on the webpage are very large and when the content and the format of the source webpage are likely to be changed [24]. Wrappers are not efficient because programmers have to find the reference point and the absolute tag path of the targeted data content manually. This requires one wrapper for each website since different sites follow different templates. The effects are increased time consumption and effort from the programmer.

4. Application Programming Interface (API)

IBM has defined an application programming interface (API) as "*a functional interface supplied by the operating system or a separately orderable licensed program* *that allows an application program written in a high-level language to use specific data or functions of the operating system or the licensed program*" [79], i.e. API is a software-to-software interface.

In the thesis experiment domain, an API allows third-party software developers to access any web services provided by developers of OSN as open source software does. Meanwhile, the privacy of the application's source code is protected as a closed application does. The power of an API is in its ability to integrate and interoperate different applications and tools with each other regardless of which programming language is used to write the application or how it was designed.

Mostly when using APIs, the developers return the results in a common XML file format or as Java Script Object Notation (JSON) objects. Table 2.3 summarizes some of the similarities and differences between them.

2.3.4 Evaluation

There are some measures for the common performance of software retrieval systems [17](p. 62-64) which include:

- Functionality: does the system satisfy all the functions that the system is designed for.

- Response time of extracting data.

- Space requirements: the developers have to balance between these two measures because there is usually a trade off between them.

- Quality: the hardest measure for the retrieval system is to evaluate the quality of the retrieved information. This is due to the vague of the user needs.

- Precision and recall measure: this is the most popular measure. The precision is used to get the accuracy of the retrieval system while the recall is the

	XML	**JSON**
Stands for	Extensible Markup Languge	JavaScript Object Notation
Extended from	SGML (Standard Generalized Markup Language)	JavaScript
Developed on	1996	2011
Developed by	World Wide Web Consortium	Douglas Crockford
Official website	http://www.w3c.org/TR/rec-xml	http://json.org
Speed		✓
Simplicity		✓
Extensibility		✓
Interoperability	✓	✓
Openness		✓
Human readable/writeable	✓	
Machine readable/writeable		✓
Resource(CPU/Memory) utilization	✓	
Provide Structure to data	✓	✓
Ease of creating data on server side	✓	
Processing easily form client side		✓
Self description data	✓	✓
Data exchange format		✓
Document exchange format	✓	
Mapping to Object-Oriented program		✓
Internationalization	✓	✓
Adopted by industry	✓	
Ease of debugging and trouble-shooting on server side	✓	
Ease of debugging and trouble-shooting on client side		✓
Lack of Security	✓	✓

Table 2.3: XML vs JSON

percentage of the actually retrieved documents. The higher precision means more probability to retrieve relevant documents while the higher number of the recall implies that the system retrieved the most relevant document from the collection.

- F- measure: it is derived from precision and recall measure. It takes the harmonic mean of the two values. i.e. the ratio of the multiplication of the precision and recall divided by the arithmetic mean.
- Average precision: is represented by a weighted precisions sum:

$$Average\ precision = \frac{\sum_{i=1}^{N} precision(i) \times relevance(i)}{|R|}$$

where R is the set of the known relevant documents. The relevance value is 1 when the document is relevant and 0 if it is not.

2.3.5 Section Summary

The first section of this chapter presents a brief overview of the "web" as it is the general domain of this thesis. It starts with a short description of the web history then it clarifies the difference between the IR, IE, DR and DE to be accurate in time of choosing the appropriate techniques to extract data from the web. Thus, it would be important to know how data is represented on the web in first place in order to be extracted. Consequently, different strategies for how to seek information on the web are described followed by various techniques of DE from the web.

The well understanding of these topics helps in determining the appropriate technique for this thesis which focuses on extracting unstructured and semi-structured data from OSN. The data extracted will be evaluated using some of the measured stated above e.g. response time.

The next section of this chapter will focus on OSN, one of the most challenging area of the web in terms of the massive changes in its data amount and speed.

2.4 Online Social Network

Section 2.3 presented a brief overview of the general domain of the web as a source of online data to be extracted. This section now focuses on a particular domain which contains a massive amount of personal information, namely that of Online Social Networks (OSNs), in order to create an overview of the nature of which forms the context of the data extraction process. What makes OSN unique from other areas of the web is that the web in general is organized around content, while OSNs are organized around users [102]. Sociologists expect that the structure of OSN will increasingly reflect the relationship of real life society [44].

2.4.1 What is an Online Social Network?

In the 1950s, Professor Barnes coined the term "Social Network" and identified its size as a group of 100 to 150 people [5]. With the emergence of the web and with the ease of communication between its users, the virtual world on the web is attracting users rapidly as it offers them the opportunity to make new friendships, to share their interests even with unknown people, and to upload photos and distribute their personal information.

Accordingly, the term and definition of OSN has expanded. Lenhart and Madden define OSNs as the "*online place where a user can create a profile and build a personal network that connects him or her to other users*" [90], while Boyd and Ellison in [41] specify three characteristics in their definition of social network sites, describing such sites as: "*social network sites as web-based services that allow individuals to:*

1. *construct a public or semi-public profile within a bounded system.*

2. *articulate a list of other users with whom they share a connection.*

3. *view and traverse their list of connections and those made by others within the*

system".

2.4.2 Brief History of Online Social Networks

OSNs started to be used publicly for communication purposes through Friendster[1] in 2003, although the concept of offline social network had been known since the 1960s. Within a few months, Friendster had attracted more than 5 million users, as reported by Mika et al. in [100] and inspired the creation of other OSNs. In Figure 2.6 which is modified from [41], a brief timeline of the launch dates of major OSNs sites is described, starting with SixDegrees.com[2] in 1997 and progressing with Facebook, Twitter and Windows Live Spaces[3] in 2006.

2.4.3 Why are Online Social Networks Important?

The popularity of OSNs has increased significantly over the last decade, coming to be at the heart of some web sites. Figure 2.7 shows the top OSN according to [78] based on the United States market share of visits. The OSN has some potential benefits which include:

1. **An OSN allows interests and trust to be shared**: users are likely to trust other people in their friends list and have common interests with each other. This attracts the development of many research systems, in order to make use of this trust [102], as will be shown in Chapter 3.

2. **OSNs influence youth**: the appropriate use of OSNs can help teenagers to be more confident in communicating with other people online as well as in keeping in touch with family members who live far away. They help people to become familiar with and up-to-date on new technologies. Schools and universities

[1] http://www.friendster.com/

[2] http://www.sixdegrees.com/

[3] http://spaces.live.com

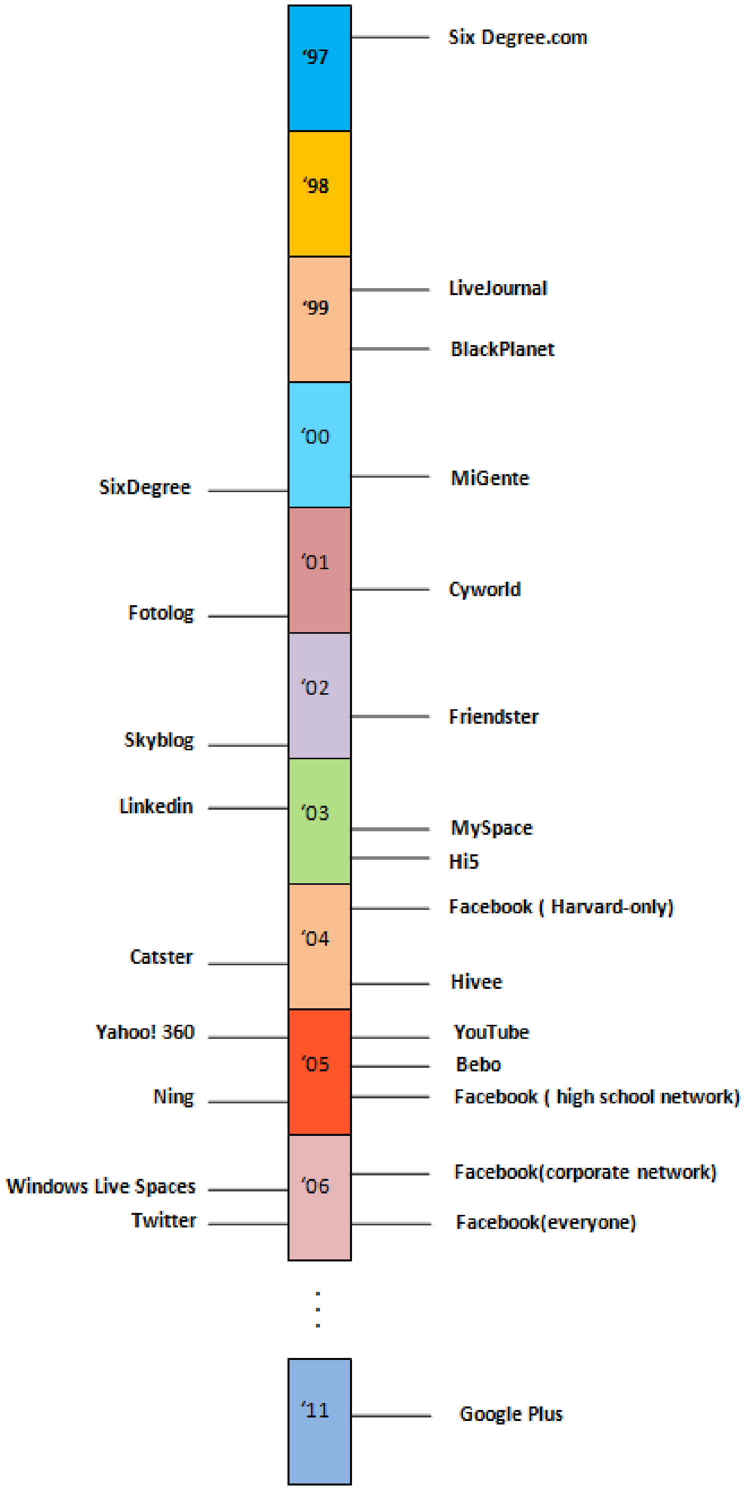


Figure 2.6: Timeline of the Launch Dates of Major Social Network Sites Modified from [41]

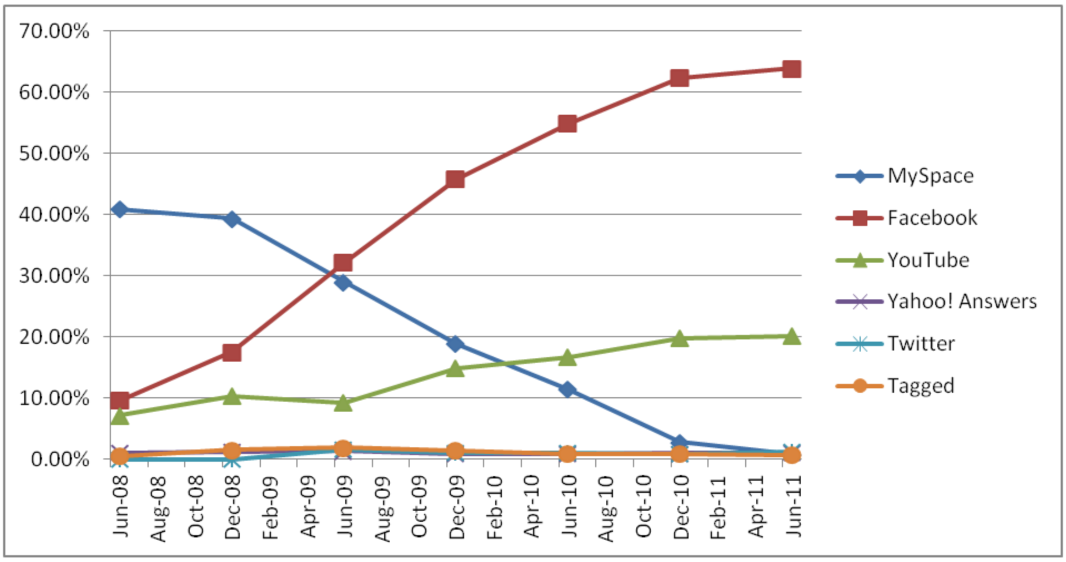


Figure 2.7: Top Online Social Network (Source of Data from [78])

encourage students to participate in discussion forums, create class blogs, explore interesting topics, and create a positive self-image on their profiles [127]. This helps them in building their identities.

3. **OSNs influence the future of the Internet**: the popularity and bandwidth-intensity of OSNs have a significant impact on the Internet traffic. Thus, understanding the structure of OSN helps in understanding the security and robustness of distributed OSNs and their impact on the future Internet [102].

4. **OSNs influence other disciplines**: besides the impact of OSNs on different fields of computer science, the OSN affects various sectors of society. Sociologists can study the behaviour of users and test their theories offline on the extracted data. Election candidates can reach a larger number of voters to deliver their programs and compete with their rivals [26]. Marketing has the largest share of benefit to gain from OSN. Governments and police can find OSNs useful for crimes associated with threats and abuses. Finally, but not least importantly, parents and social workers can monitor children's and

students' behaviours.

2.4.4 Dangers of Online Social Networks

However, as with any new technology, the rapid expansion of OSNs contributes to rising worries for parents, sociologists and governments, especially when the youth represents the heart of OSN sites activities, with little control or protection for such online interaction [126, 41]. Businesses, job seekers and adults should also be aware. Some of the dangers include:

1. **Privacy**: breach of privacy is the main issue accompanying use of an OSN. This breach could be either by other users in the friends list who could comment on your private information, e.g. mobile number, address or date of birth, or by the third party companies which share the users' information through OSN applications.

2. **Phishing / Scams**: when data becomes a currency, it is more likely to be stolen or used for potential crime such as identity theft or fraud.

3. **Children / Students**: although most OSN sites require a minimum age limit to create a new profile, fooling the system is too easy. Children tend to post detailed information about their identities e.g. phone number, school name, class schedules, etc., and this could allow them to be stalked by strangers who claim to be as the same age as them and to have the same interests. Posting inappropriate information or pictures may land children in trouble due to violation of institution policy.

4. **Employment**: job seekers should be aware of what they post on their profile. They should expect that employers are likely to make a background check on their OSN accounts in order to gain a reflection of their personal character.

Serious cases have been reported where contracts have been terminated due to a picture or a statement posted on OSN profiles.

2.4.5 Section Summary

The second section of this chapter has given a brief overview of OSNs as the specific context for data extraction in the thesis. The OSN is selected as it is one of the most recent areas of public information which provides a wealth of rapidly updated personal data. Despite its advantages, the OSN may also pose a threat; especially for youth, who represent the heart of most OSNs. The next section describes MAS, one of the techniques which could be used to improve the process of extraction and monitoring of OSN data.

2.5 Agents and Multi Agent Systems

With the rapid growth in the technology of the software industry, many expectations which were formerly considered impossible have become more achievable. For example, users previously gave detailed instructions to the computer in order to fulfil a specific task. The demand for the development of approaches which allow computers to think and act toward a predefined goal on the individual's behalf and without human intervention has grown over time.

This idea has in fact become the cornerstone in the emergence of software agents (or agents) for short as an Artificial Intelligence (AI) field. Agents are a combination of several fields in AI: encompassing knowledge representation, reasoning and learning [32](p. 181). The following subsections present the basic principles of agents and their history, features, classifications and environment, as well as considering MASs, which are a collection of agents.

2.5.1 Brief History of the Agent

Distributed Artificial Intelligence (DAI) is considered to be the root of the Software Agent since it contains, besides Distributed Problem Solving (DPS) and Parallel Artificial Intelligence (PAI), MAS from which the Software Agent has evolved, as will be explained. This means that the Agent has inherited some features, good or bad, from DAI and AI, such as: modularity, speed, reliability, reusability, platform independence and ease of maintenance [108].

Most researchers including [105, 118, 7, 15] have traced the proponent of agent technology back to the 1970s, and in particular, in 1977 when Carl Hewitt introduced his Actor Model and termed it “éactorí” [74]. He defined his actor as “*computational agent which has a mail address and a behavior. Actors communicate by message-passing and carry out their actions concurrently*” [108].

In contrast, Alan Kay(1984) traces agents' history to two decades earlier in the mid-1950s when John McCarthy originated the idea of an agent, and then his colleague in the Massachusetts Institute of Technology, Oliver Selfridge, coined the term “agent” a few years later. Together, they proposed a system that “*when given a goal, could carry out the details of the appropriate computer operations and could ask for and receive advice, offered in human terms, when it was stuck. An agent would be a “soft robot” living and doing its business within the computer's world*” [42](p. 4).

Hyacinth Nwana in [108] has split the research on agents into two main strands. The first strand, which started in 1977, worked on smart agents and concentrated on macro issues besides the theoretical, architectural and language issues. The aim was to specify, analyze, design and integrate systems comprising multiple collaborative agents. Some classic systems and work which resulting from this includes the actor model [74].

The second strand started in 1990. It focused on a "*much broader range of agent type, from moronic to the moderately smart. The emphasis has subtly shifted from deliberation to doing, from reasoning to remote action*" [108, 42](p. 1-4).

2.5.2 What is an Agent?

Although researchers trace the concept of the agent to almost 40 (or 60) years ago, to date, there is still no approved definition of a software agent [28, 48, 129].

One of the reasons mentioned by Nwana as to why the definition is sensitive is that the term "agent" is an umbrella for various fields of information technology research hiding highly diverse accepted synonyms [108, 61].

Thus, in the absence of a universal consensus on the definition of an agent, end-users think of the agent as "*the program that assists people and acts on their behalf*" [88], while software developers and software engineers count agents as a functional program which depends on the input-computer-output operational structure, although many systems have a reactive flavour since they must maintain ongoing interaction with their environment [37]. From the perspective of AI researchers, agent means "*an abstract entity that is able to solve a particular problem*" [107].

However, the definition chosen for the current thesis is that provided by Wooldridge and Jennings in [130], due to its greater precision in describing the system developed, as well as the fact that it is increasingly adopted by many researchers in the field [28]. The definition states that,"*An agent is a computer system situated in some environment and capable of an autonomous action in this environment in order to meet its design objectives*".

Despite the multiplicity of agent definitions, all definitions agree that an agent is a special piece of software which acts as a human agent to achieve its client's agenda with consideration of autonomy, as will be detailed in the upcoming sections [30].

The lack of clarity in the definition of agents leads to some confusion with the

Features	**Classic Expert system**	**Agent**
Acting with the environment directly	x	✓
having reactive and proactive behaviour	x	✓
sociability, ability to cooperate, coordinate or negotiate	x	✓

Table 2.4: Comparison between Agent and Expert System

different sectors of workers in the field of Information Technology (IT). For example, AI researchers consider the agent as an expert system due to its ability to think and decide to act.

While this may be true however, the agent has extra features that the classic expert system does not [129](p. 27-28) such as those which are shown in Table 2.4. Another example includes most programmers who are familiar with Object-Oriented languages, who commonly confuse agents with objects. Table 2.5 summarizes some of the similarities and differences between agents and objects [129](p. 25-27), [6].

Object	**Agent**
High level of software abstraction	
• Defined in terms of classes, attributes and methods. • Difficult to write code of actions. • Less autonomy:“the decision lies with the object that invokes the method”. • Some page structure is replicated. • Difficult to update contents.	• Defined in terms of its behaviour.. • Easy to specify behaviours. • More autonomy: decides to perform an action or request another agent to do it.“the decision lies with the agent that receives the request”. • Has flexible behaviour(reactive, social, proactive). • It is multithreaded. Each agent has at least one thread of control.

Table 2.5: Comparison between Agent and Object

2.5.3 Why are Software Agents Important?

> *"As future generations of computing systems begin to sound more like biological systems than mathematical ones, we believe software agents will play a central role in network-intensive applications, self-healing systems, adaptive software applications, and software systems in general"*
>
> **Qusay Mahmoud [97]**

In general, the importance of the software agent comes from two aspects [42, 6]:

1. The need to simplify the efficient description and building of complex systems for different applications, especially those applications that involve distributed computation, communication between components, sensing or monitoring environments or autonomous operation.

2. The need to overcome user interface problems: e.g. the interface agent can help users in completing forms from different websites, such as for booking a flight, that have different fields.

As a result, the main software development companies which use object technologies are motivated to increase their interest in agents: Oracle[1] for example considers the use of an agent approach as a reliable solution for defining a new "client-agent-server"[2] architecture [63]. These companies are motivated for three reasons, as [61] states. Firstly, motivation stems from the belief of AI researchers in the advantages of working with simpler, more flexible and reusable software entities such as software agents to overcome the disadvantages of AI which for many years have led to complex and hard to develop software.

[1] http://www.oracle.com

[2] "The client-agent-server is an extension of the client-server model where the agent plays a central role for dispatching client requests on already registered servers" [131]

The second factor is the appearance of the necessity of having autonomous and mobile objects to be used in designing new object oriented languages especially after the integration of the object world in the network environment.

Finally, a new direction in computer use has appeared via the expansion of networks and distributing of resources among different various systems. This leads to the use of agents to automate control and management processes, and to quickly adapt the network's behaviour to the client's needs.

2.5.4 What is a Multi Agent System (MAS)

The agent cannot be found isolated in the environment. Each agent has a limited viewpoint, incomplete information and each has a sphere of influence. By working together with asynchronous computations, they can perform the tasks that are considered difficult or impossible for an individual agent to solve more efficiently. Just like people, each agent is specialized for a particular task. The "*collection of software agents that communicate and cooperate with each other*" [6] to accomplish a common goal is termed a Multi Agent System (MAS).

The environment or the world of software agents can be represented by computer operating systems, databases or networks. Some agents live in an artificial environment such as a computer screen or in its memory. Each agent in MAS has to sense its environment and act autonomously upon it according to its own agenda [6, 60].

2.5.5 Classification of Software Agent Features

Based on the different definitions of agents as described in Section 2.5.2, identification and classification agent features differ according to the field that the agent has been used in and its behaviour [59]. Moreover, the same feature may be known by more than one name. Some of the agent features are explored as follows:

1. Autonomy

The agent has the ability to operate as a standalone process, which means it can operate independently without the direct intervention of users. In other words, it exerts control over its actions and internal state. For instance, provided by the user with the goal and plans of a specific task, the agent must make a decision on how to achieve the goal in a reasonable manner. This can be achieved due to the fact that each state of the agent's behaviour is done for a purpose.

2. Learning (Adaption)

The agent has the ability to learn from its environment and/or users. It uses its previous knowledge then combines it with the data currently detected from the environment. Consequently, its reaction is based on the experience that it has to achieve its goals. The agent learns via different mechanisms such as:

- using a process known as machine learning, whereby the machine can improve its performance based on previous results.
- exchanging metadata (data about data).
- using knowledge discovery and data mining, which are interdisciplinary areas. Knowledge discovery and data mining are methodologies for automatically extracting patterns which could be considered as knowledge about data from large volumes of data. [82, 4].

3. Sociability

As stated earlier, an agent is designed to perform a particular task. Mostly the agent has to communicate with other agents; users, other software agents or other software processes, by means of some kind of language. Communication can either be direct, provided agents speak the same language, or indirect through use of an interpreter, provided agents know how to talk to the interpreter and vice versa [33](p. 186). Sociability does not mean exchanging bytes, it means the ability to communicate at the knowledge level, i.e. exchange their beliefs, goals and plans

4. Cooperation

This feature is built on the previous feature (sociability). It was stated in Section 2.5.3 that one of the points which makes the agent important is the need to simplify complex systems through splitting tasks into smaller pieces of tasks. Allocating an agent to each task requires the ability of theses agents to cooperate with each other to achieve the common goal.

An agent uses messages to communicate and cooperate with other agents, humans, or application programs. Each message contains a set of attributes, as will be detailed in Section 6.2.4 and Appendix (C). Design of messaging format and communication protocols are part of the basic agent mechanism and are not any more the responsibility of the system developers [6].

5. Perceptivity

This feature is included implicitly in most definitions of agent. It involves detecting the changes in the environment, whether the physical world, Graphical User Interface (GUI), Internet or a combination of all of these, then responding or taking action in such a way as to achieve an effective balance between the goal-directed designing agent and reactive behaviour.

6. Mobility

The mobile agent is a promising technology due the simplicity it brings to the design and implementation of different systems such as distributed systems and network management. It can travel between different nodes in a computer network without the need for a continuous connection between machines [33](p. 20).

Mobile agents operate autonomously and asynchronously without a need for monitoring by the user, which saves time and reduces communication costs. Also, mobile agents can go offline when the network connection is too busy [115].

However, the mobile agent suffers from security issues as reported in [65] and [114](p. 267). The mobile agent may harm the host or vice versa in a MAS with a low level of security.

7. Proactivity

The agent has the constantly spirit of initiative to achieve the goals that are delegated to it, not simply act in response to its environment [103].

Other features:

Agents are characterized with some other features such as: truth; an agent will not provide false information intentionally, benevolence; it tries to perform its task, rationality; it will never prevent itself from achieving its goals [30]. Table 2.6 summarizes some of the agent's features.

The most accepted classification for an agent based on its features is that presented by Nwana in [108]. He classified agents into three main categories based on its features: cooperative agents, autonomous agents and learning agents. Overlap between two of these features creates other features: collaboration agents, interface agents and collaboration learning agents respectively, as shown in Figure 2.8. If all these six features are combined in one agent, then the agent is termed a smart agent.

Franklin and Graesser in [60] summarize some of these features along with their synonyms, as shown in Table 2.7. Despite this diversity, there is a minimum set of common features that all agents share [37, 30].

Feature	**Description**
Autonomy	The ability to operate as a standalone process to achieve the goals and perform actions such as task selection, prioritization and decision-making with some kind of control without direct intervention of user.
Sociability	The ability to communicate with other agents (users, software agents, software processes) by some kind of communication language at knowledge level by exchanging their beliefs, goals and plans.
Perceptivity	The ability to detect changes in the environment and respond in a timely fashion in such a way that achieves an effective balance between goal-directed and reactive behaviour .
Proactivity	The ability to exhibit goal-directed behaviour by taking initiative, not just act as response to the environment.
Mobility	The ability to convey itself between different machines in the network.

Table 2.6: Agent Features

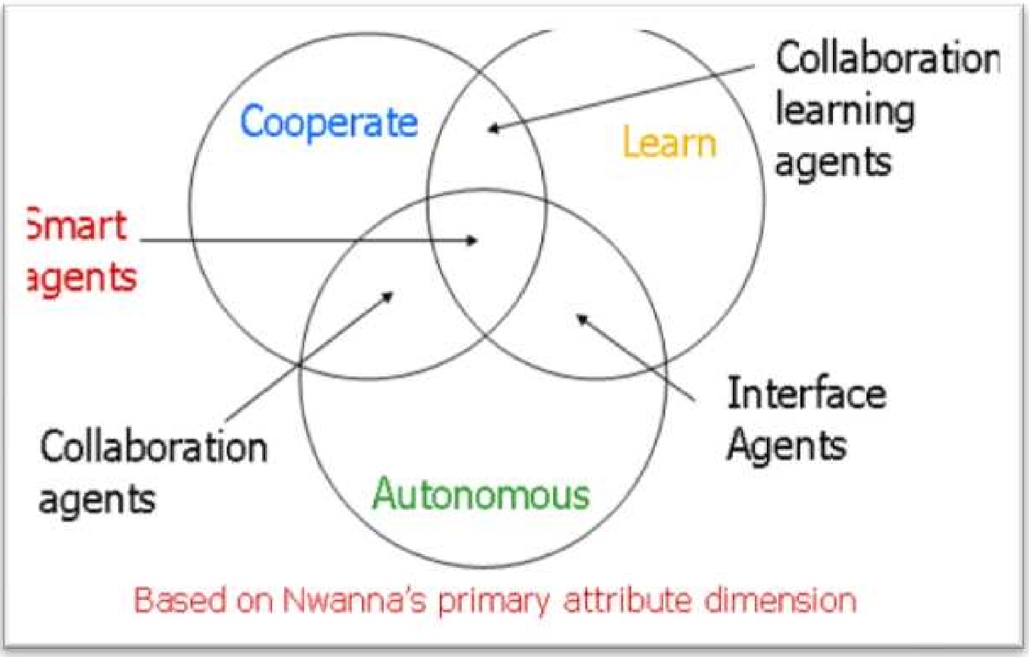

Cooperate
Learn
Collaboration learning agents
Smart agents
Interface Agents
Collaboration agents
Autonomous
Based on Nwanna's primary attribute dimension

Figure 2.8: Nwanna's Primary Attributes Dimension

property	Other names	Meaning
reactive	(sensing and acting)	responds in a timely fashion to changes in the environment.
Autonomous		exercises control over its own actions.
Goal- oriented	pro-active purposeful	does not simply act in response to the environment.
temporally continuous		is a continuously running process.
communicative	socially able	communicates with other agents, perhaps including people.
learning	adaptive	changes its behaviour based on its previous experience.
mobile		able to transport itself from one machine to another.
flexible		actions are not scripted.
character		believable "personality" and emotional state.

Table 2.7: Classification of Agent Features by Franklin and Graesser

.

2.5.6 Agent Programming Languages, Platforms Frameworks

Although many languages can be used to design and implement agents, such as Java[1], C/C++, Lisp, Prolog, Tcl, Smalltalk[2] and Perl[3], it is very difficult to build an agent from scratch. This would require sound experience in different areas such as "agent architecture, communications technology, reasoning systems, knowledge representation, agent communication languages and protocols, and machine learning" [71](p. 216).

The selected language for building the agent must support agent features, database and knowledge base, mediated architecture (i.e. it is able to add information to and subtract information from a computer), support multi-threaded and finally support communication and execution security. Mobile agents require a language that sup-

[1] http://www.java.com/

[2] http://www.smalltalk.org

[3] http://www.perl.org/

ports mobility [6]. Bordini et.al in [36] classify agent programming languages into three categories:

1. **Declarative languages**: which are "partially characterized by their strong formal nature, normally grounded on logic" e.g. CLAIM, FLUX, Minerva, Dali, and ResPect.

2. **Imperative languages**: which are "purely imperative approaches to agent-oriented programming". This type is less common than the previous one as agent-oriented design is declarative in nature. An example of an imperative language is JACK Agent Language (JAL).

3. **Hybrid languages**: various agent programming languages are hybrid. Such languages are declarative as well as providing some specific constructs allowing for the use of code implemented in some external imperative language. Examples include 3APL, Jason, IMPACT, Go!, and AF-APL.

Note that agent framework implementation is not strongly tied to the programming language, but is concerned more with supporting agent features such as communication and coordination [36]. Some agent construction toolkits (AgentBuilder[1], Aglets[2] and JADE[3]) which were either used or were tried out and rejected in the building of the MAS for the current study will be described briefly in Section 6.2.1 as well as in appendices (B) and (C).

2.5.7 Section Summary

This section clarifies the definitions of agent and MAS, which are selected as a technique to improve data extraction from OSNs. Some features of the MAS are

[1] http://www.agentbuilder.com/

[2] http://aglets.sourceforge.net/

[3] http://jade.tilab.com/

explained to show how these features will fit with the requirements of OSN data extraction. Selection of programming language is critical in designing a MAS due to building a MAS from scratch is time- and effort- consuming. Furthermore, it requires a broad knowledge base of several domains (e.g. mediated architecture, database and knowledge base).

Since it is well-known that MAS is a complex system to design, the next section presents a brief overview of the formal specification to be used before building MAS.

2.6 Formal specification

Developing a software application requires specifying in detail how to solve the problem either in formal or informal specification. This section presents a brief overview of the definition of formal specification, its importance and some description Object-Z language, which is an extension of one of Z notation, the most common language for formal specification.

2.6.1 What is Formal Specification

Ratcliff in [116](p. 13) defines formal as a "*notation [which] enables us to describe the outward form or essence of something in a precise and clear way*", while Currie in [53] describes specification as "*a statement of requirements for a system, object or process*". Based on these and similar definitions, several definitions have been stated to define the formal specification.

Duke and Rose define formal specification as "*the process of creating precise (mathematical) models of a proposed system. The role of such a model is to provide unambiguous description of the functionality of the system*" [55]. In [27], Baryannis classifies 9 definitions of formal specification according to the domain. The definition that fits with the OSNRS in this thesis is the one which is in the context of Web services. It identifies the formal specification as "*a precise mathematical description, supported by a sound logic and reasoning apparatus, of what the service should do in terms of its capabilities and the functionality it offers*".

2.6.2 Why Formal Specification

Formal specification describes "*what the system should do, not (necessarily) how the system should do it*". The importance of formal specification can be viewed via the drawbacks of informal specification. Using natural language, as with informal specification, may be satisfactory in some cases of specifying software due to natural language is expressive and does not require training [38], However, as George suggests in [64] cited from [69], using natural language leads to three problems:

1. *Ambiguity*: the meaning and interpretation of a word in natural language may differ according to its context. This vagueness is considering as a drawback in software development.

2. *Incompleteness*: the details may be unspecified and incomplete in natural language.

3. *Contradiction*: the system requirements may contain conflict or inconsistency between statements.

Moreover, if the software is considered to be one of the more complex systems to develop, as is the MAS which forms one of the developed approaches in the thesis, it would place extra pressure on software developers when they decide to use MAS [134, 76]. Thus, they must avoid issues such as running over time, exceeding the budget, producing incorrect or inefficient software products and not being able to reuse the software components. Mostly, these problems are caused as a result of choosing informal, imprecise methods at different stages of the software engineering process [53, 25].

In brief, using natural language alone to design a MAS is not sufficient. Natural language is inaccurate and can tolerate more than one interpretation [112]. Usually, this is the main cause of the production of ambiguous and incomplete system specifications.

2.6.3 Formal Specification Languages

Saaman et.al. in [119](p. 11) describe a formal specification language as a programming language in which operations definitions are not be executable as they cannot be compiled or interpreted. Formal specification languages share a similar general structure although they vary widely in detail.

There are many formal specification languages which have been developed through the years. The next section presents just one of these languages, the Object-Z, which is used in the implementation of the MAS developed for this thesis. Object-Z supports the required features of MAS, specifically; concurrency, communications and state.

2.6.4 Object-Z

Object-Z is an object-oriented extension of the Z notation, a leading formal specification language, in which all syntax and associated semantics of Z are part of Object-Z. Object-Z was developed by researchers at the University of Queensland to facilitate the specification of Z in order to overcome the scalability and structural problems of formal languages [64, 55].

The general structure of Object-Z [53](p. 32), [55](p. 6-12) is shown in Figure 2.9. The structure contains a class which combines (in order):

- **Visibility list**: specifies the interface of the class objects to be visible to the

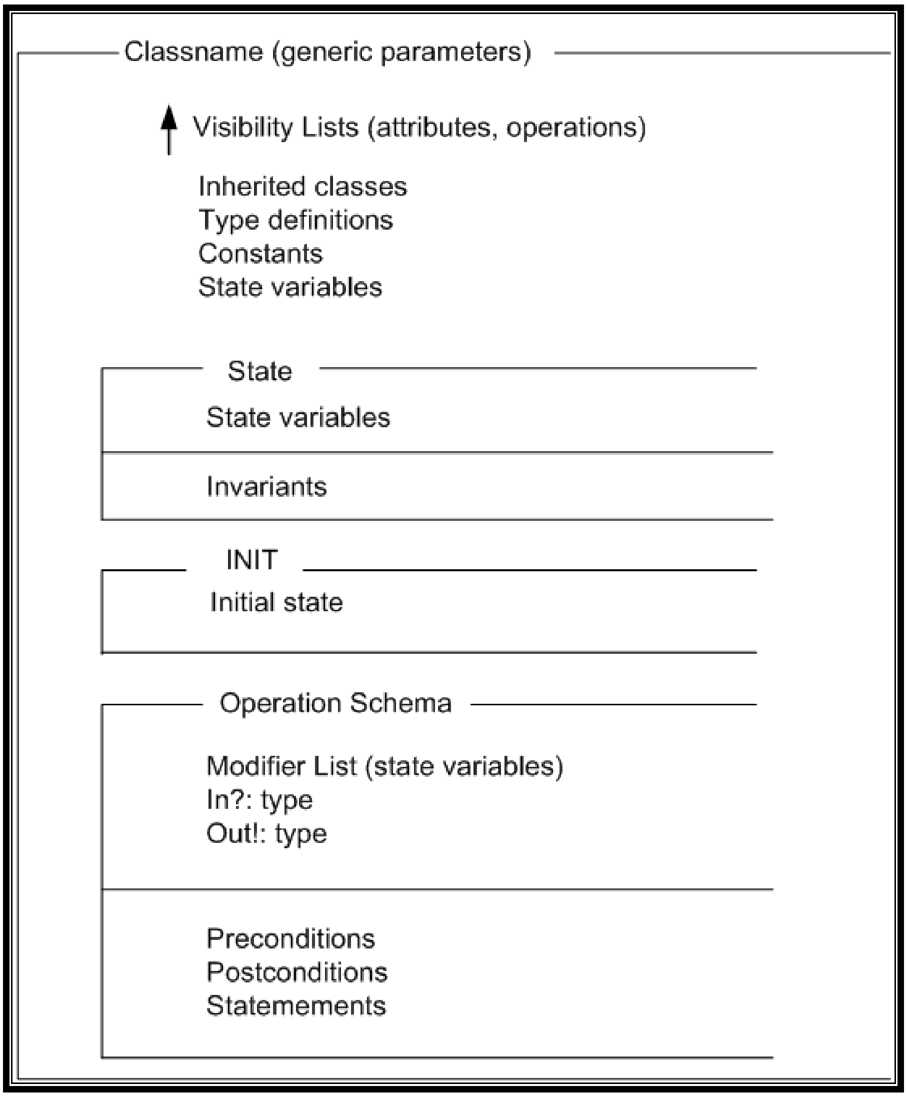


Figure 2.9: General Structure of Object-Z Class Modified from [64]

class environment. If there is no explicit visibility list, then it considers that all features of objects are visible.

- **Definitions of types, variables and constants**: defining the values which types, constants and variables may take.

- **State schema**: unnamed box to declare all information related to the class. It is divided into two parts: the upper part contains the declaration of the class state variables while the lower part (termed the class invariant) contains the predicate involving the constraints on the values of state variables declared.

- **Initial schema**: always called *INIT* which consists of a predicate conjoined with a class invariant to define the initial condition. Note that the initial schema is not just an operation of initialization; it could involve returning the object to its initial configuration by one of the class operations during the evolution of the system.

- **Operation schemas**: involve a set of communication variables for passing messages between the class objects and their environment. It specifies the relationship between the variable values before and after executing the operation [55](p. 10).

Initial Object-Z supports defining class unions outside inheritance restrictions. Also multiple communications can be applied through a range of composition operators [40]. Appendix (D) contains some of the Object-Z notations which are used in Chapter 5.

2.6.5 Section Summary

The last section of this chapter defines the formal specification and explains its importance in specifying the system at the design stage to minimize the time and cost of the implementation stage. The section focuses on Object-Z language as it is used to formally specify the application developed in this thesis.

2.7 Chapter Summary

The aim of this chapter was to present a brief background to the three main disciplines of the thesis which are: web DE, OSN and MAS. Also, several relevant topics such as formal specification have been defined and their importance explained.

The chapter started by presenting the general domain of the thesis, the web, and how data is represented and extracted from it using different techniques. The focus was then narrowed to OSNs, one of the hottest areas on the web. The OSN was selected due to it being distinctive in its wealth of continuously updated personal data over time, provided by users rather than documents.

Extract data from OSNs and monitor updates over time requires technology such as MAS to support the traditional techniques for web DE. Thus, the chapter presents a brief overview of agent, MAS and some features of which fit the requirements of the proposed algorithm. However, MAS is considered a complex system which needs to be specified formally to ensure that the system is robust before the implementation stage. Therefore, an overview of Object-Z structure is presented as the selected language to formalize the proposed MAS for DE from OSN.

Next chapter presents state of the arts in using some of these techniques in DE from web in general and OSN in particular.

Chapter 3

Literature Review

3.1 Introduction

The huge growth in data available on the web increases the demand for methods and tools to process and extract this data. Different approaches to the search for information on the web have been presented in the previous chapter. Figure 3.1 shows the areas which will be covered in the literature review of the thesis.

The rest of the chapter is organized as follows: Section 3.2 introduces previous works related to different approaches for extracting data from the web in general, while Section 3.3 narrows the research scope to the approaches for extracting data from OSN. Section 3.4 explores research using MAS technology in seeking information from different domains on the web. Section 3.5 describes state of the art research similar to this thesis in extracting data from OSN using MAS. Section 3.6 illustrates some of the formal specification languages which are used to formally specify MAS. Finally, Section 3.7 summarizes the chapter.

3.2 Approaches for Web Data Extraction

A considerable amount of literature has been published on the various approaches for seeking information on the web. This section explores some of the research on extracting data from web pages. Search engines, as the most well-known approach for

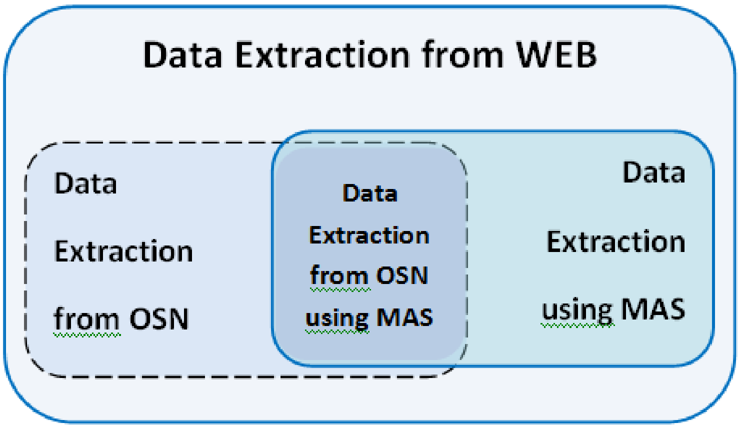


Figure 3.1: Division of Areas Covered in the Literature Review

web DE, ignore at the time of indexing extra contents of web pages which surround the main content such as advertisements, headers, footers, copyright information navigation links, comments and reviewing. As mentioned in Section 2.3.2, search engines rely on web crawlers to extract data. The core technique to construct the appropriate web crawler is the wrapper [83], which identifies data of interest and maps them to suitable formats.

A leading study presented by Laender and Ribeiro-Neto in [87] surveys around 15 different web data extraction tools in order to provide a qualitative analysis of these tools. The tools have been characterized into 6 groups based on the main technique used by each tool to generate a wrapper, as follows:

1. *Language for Wrapper Development*: specially designed languages to help users in constructing wrappers.

2. *HTML-aware Tools*: these inherit HTML documents' structural features through turning the HTML document into a kind of representation which reflects the tag hierarchy of the HTML.

3. *NLP-based Tools*: using Natural Language Processing (NLP) techniques (e.g.

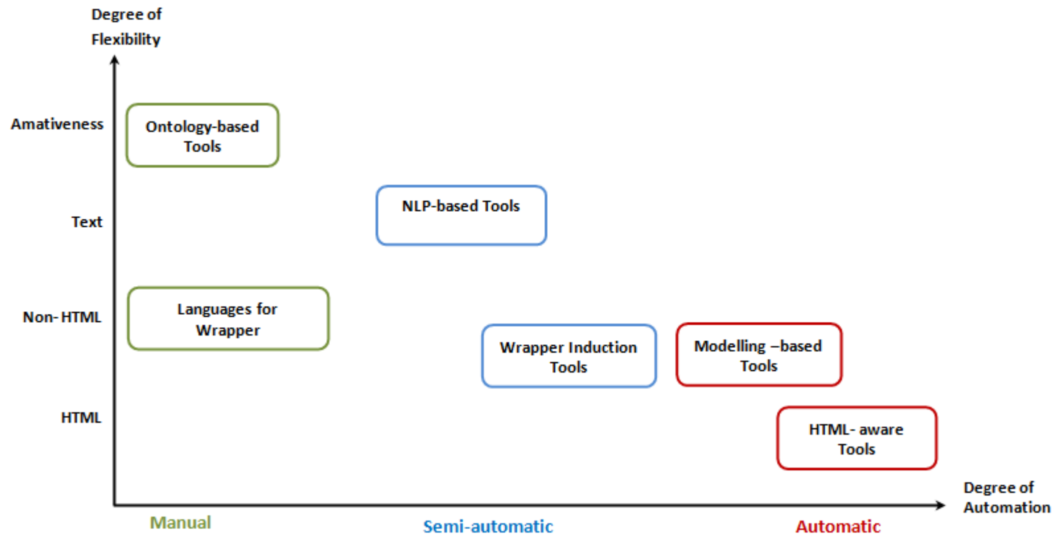


Figure 3.2: Qualitative Analysis of DE Tools Modified from [87]

filtering and part-of-speech tagging) to correlate the phrase and sentence elements to learn rules for DE.

4. *Wrapper Induction Tools*: generate rules for extraction based on formatting features which delineate the structure of data pieces found in a given set of training examples. These tools are independent of linguistic constraint, which quality distinguishes them from NLP-based tools.

5. *Modeling-based Tools*: locate portions of data which implicitly conform to a given structure of interested objects.

6. *Ontology-based Tools*: these are unlike all previous tools, which rely on the structure of the data within a document. Rather, the extraction of these tools relies on data directly through locating constants present in the page in order to construct the interested objects.

Laender and Ribeiro-Neto summarize a qualitative analysis of each tool in Figure 3.2. A more recent study presented by Amin et al. in [21] classifies research on wrapper generation into three categories:

1. *Wrapper Programming Languages*: these are specialized pattern specification languages which aim to help users in creating an extraction system.

2. *Wrapper Induction*: this is created to overcome the data manual labeling of web pages contents. It learns data extraction rules from examples which are manually labeled.

3. *Automatic Extraction*: these are techniques to generate a web wrapper automatically.

Table 3.1 presents some studies which develop different DE tools. The first two studies [21, 24] develop wrappers based on HTML tags in order to find the primary contents of the web pages, e.g. <`TABLE`> and <`DIV`>. The structure and layout of web pages differ according to the content type. Consequently, the position or the tags of main contents also differs. Thus, the second two studies [62, 23] propose a tag independent algorithm and rely on a DOM tree.

Table 3.1: Different Methodologies of Web Data Extraction

Research study	**Advantages & Disadvantages**
[21] 2009 The study concentrates on extracting tabular data patterns from a web page in a linear time. The data came from back end database (deep web). The proposed approach automatically discovers the structure of the table through finding a relevant pattern mining efficiently from web pages. It then generates a regular expression for the extraction process to implement a FastWrap application. **Methods** • Creating automated wrappers for tabular data. • Depends on HTML pages as inputs. • Using suffix tree to implement a pattern extraction.	**Advantages** The authors claim that they propose a novel technique in generating wrappers. • Fully automatic system. • Reliable and accurate wrapper. • Algorithm works in linear time to generate the wrapper. **Disadvantages** Work for structured data

Table 3.1: (continued)

Research study	Advantages & Disadvantages
[24] 1997 The study presents an approach for automatically generating wrappers for semi-structured data. The approach hypothesizes the structure of the page from the formatting information in the HTML page of the source. The system is built as follows: 1. Identify the structure of the source through: • Identifying tokens of interests. • Determining the hierarchical structure of the page using heuristics. 2. Building a parser for the source pages through: 3. Adding communication capabilities to the wrapper in order to determine the URL of the page, to make an http connection and to retrieve data. The wrapper generated facilitates querying the source and may integrate with other sources. **Methods** • Creating automated wrappers for semi-structured data. • Build a parser which depends on HTML tokens. • Using heuristics (e.g. font size and indentation) to identify the important tokens and the page's hierarchical structure.	**Advantages** The paper describes the steps for generating the wrapper in detail, including how to identify the tokens and build the parser. **Disadvantages** • The approach depends on what the authors term correction steps, whereby the user must manually correct a token or a rule in the grammar of the section hierarchy. • The system cannot identify tokens in a new unexpected format other than the one that the parser is designed for. • The web page used in the experimental work is too simple and tidy. • The system cannot extract tabular data.

Table 3.1: (continued)

Research study	**Advantages & Disadvantages**
[23] 2010 The study proposes a tag independent algorithm called the Visual Clustering Extractor (VCE) which simulates how the user behaves in visiting web pages to find the main contents through two phases: 1. Transform the DOM tree of the HTML web pages into a block tree. A block is a set of elements which has • The same visual styles and order as DOM tree. • The same conceptual purpose of page appearance. 2. Traverse the block tree recursively to find the main block. **Methods** • Approach is applicable to structured, semi-structured and non structured data. • Tag independent. • Having one iteration • Does not have learning phase. • Able to remove comments from the main contents.	**Advantages** • Build a wrapper which depends on DOM tree • For evaluation, the result is compared with a previous algorithm called K-FE.

Table 3.1: (continued)

Research study	**Advantages & Disadvantages**
[51] 2010 The study proposes an algorithm for classification and recognizing automatically either the semi-structured table on the web page as a layout table (i.e. non relational table, e.g. formatting or navigational tables), attribute/value table or other. The investigation involves 5000 HTML tables over 1.2 billion high quality English pages on the web. **Methods** • Using a crawler. • The study evaluates the results manually to find precision, recall and F-measure	**Advantages** • A large number of web pages have been crawled. • The study is able to find 79% correct subjects of the tables, which is often not present in the table itself. **Disadvantages** The research does not mention how the data has been crawled.

Table 3.1: (continued)

Research study	Advantages & Disadvantages
[49] 2002 The study discusses how to design an effective parallel crawler to maximize its performance. It presents some issues related to parallel crawlers such as: • Overlapping in downloading same pages several times. • Quality of the pages to have priority for downloading due to each crawler having a limited view of the whole web. • Communication bandwidth: the crawlers need to communicate with each other periodically to coordinate the crawling process to prevent the previous two issues (overlap and quality of downloaded pages). Consequently, the study proposes metrics for evaluation of different techniques for parallel crawlers. **Methods** • Build parallel crawlers. • Using Breadth First Search algorithm.	**Advantages** Leading study in clarifying the concept of parallel crawlers in terms of scalability (40 million pages). It presents different techniques for parallel crawlers, which were considered closely guarded secrets at the time of paper publication as the authors claim. **Disadvantages** The research does not mention how the data has been crawled.

The types of structure used in web pages as underlined in Table 3.1 include tabular data (structured data), semi-structured data and unstructured data. Tabular data is targeted by Amin et al. in [21] while Crestan et al. in [51] set semi-structured data as the target. Other research attempted to generalize the extraction method to be flexible with structured and loosely structured data as done in [23] and [111].

However, usually the crawlers cannot cover the deep web, where the majority of useful data is stored. This is because crawlers rely on hyperlinks in finding new web pages and have a limited ability to submit forms [96]. Thus, Hedley et al. in [72] generated a method to detect the templates then analyse the textual content and the document's adjacent tag structure to extract query related data.

Liu and Zhai in [92] realised the importance of extracting data records which are retrieved from deep web (backend database). They proposed a method termed nested data extraction using tree matching and visual cues (NET) to extract flat or nested data records automatically.

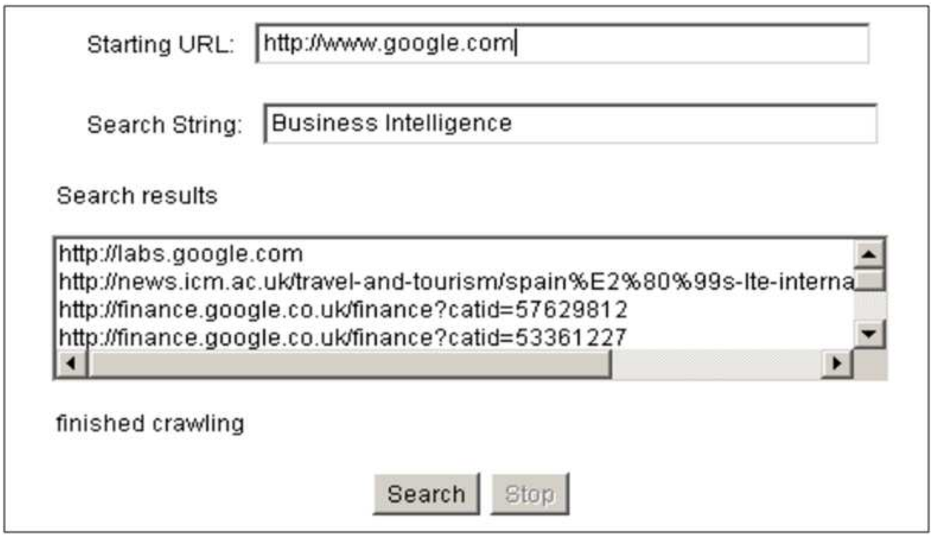
[50] demonstrated a system to grab data from the web automatically. The proposed system is also similar to two other approaches: in the first one, [84] are concerned with analysing news websites by identifying pages of news indexes and pages containing news; their work aims to classify pages according to their structure, without any previous assumptions. [46] approach is focused on crawling: in contrast to [84], this system is focused on structure recognition rather than searching for pages which are relevant to the topic. This approach does not crawl data behind forms.

Based on the above and on similar research, it is possible to classify the approach adopted for this thesis under HTML-aware tools, which is presented in [87], and as more likely to be under wrapper induction regarding classification in [21], as will be explained in the next chapter. It is similar to [24] in terms of relying on token analyzers to find the data of interest in the HTML web page.

3.2.1 Thesis primary experiments

Since web crawlers are a central part of all search engines, details of their algorithms and architecture in the past tended to be kept as business secrets. Currently however there are several open sources which explain the basic concept of how to crawl the web. The starting point for this thesis in extracting information from the web is established based on the functionality of the search engine crawler of two programs: "WebCrawler", created by Thom Blum et al. [34], and "Search Crawler", produced by James Holmes [120]

The developed application "My Search Crawler", as illustrated in Figure 3.3, intends to surf the web looking for a particular data as most search engines do. It works as follows: when the user specifies a web page and the keyword(s) of his/her interest, the application has to retrieve the hyperlinks (URL addresses) of sub web pages that contain the required keyword(s), and the results are then listed in the "Search Result" area.



Starting URL: http://www.google.com

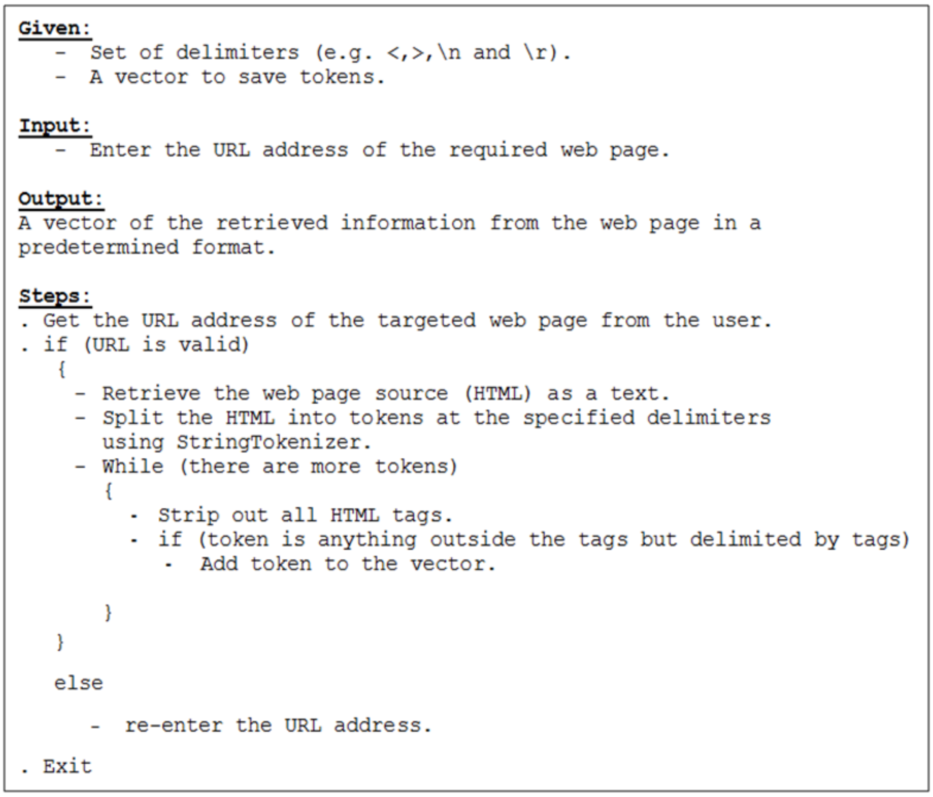
Search String: Business Intelligence

Search results

```
http://labs.google.com
http://news.icm.ac.uk/travel-and-tourism/spain%E2%80%99s-lte-interna
http://finance.google.co.uk/finance?catid=57629812
http://finance.google.co.uk/finance?catid=53361227
```

finished crawling

Figure 3.3: Screen Shot of the Output of My Search Crawler



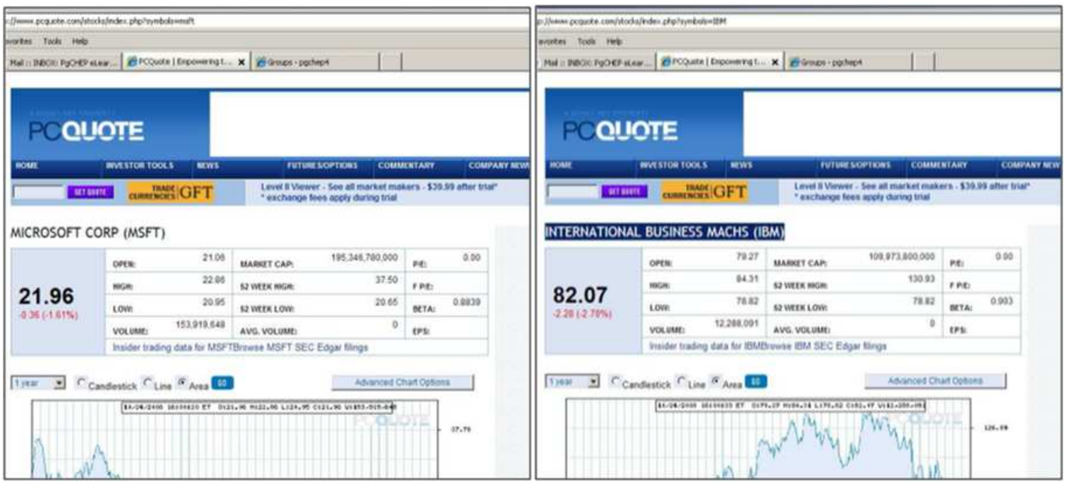
```
Given:
    -  Set of delimiters (e.g. <,>,\n and \r).
    -  A vector to save tokens.

Input:
    -  Enter the URL address of the required web page.

Output:
A vector of the retrieved information from the web page in a
predetermined format.

Steps:
. Get the URL address of the targeted web page from the user.
. if (URL is valid)
    {
      - Retrieve the web page source (HTML) as a text.
      - Split the HTML into tokens at the specified delimiters
        using StringTokenizer.
      - While (there are more tokens)
        {
          · Strip out all HTML tags.
          · if (token is anything outside the tags but delimited by tags)
              ·  Add token to the vector.

        }
    }

    else

       -  re-enter the URL address.

. Exit
```

Figure 3.4: Pseudocode of HTML Parser Modified from [68]

The second early stage in the progress of the thesis is to understand how to parse the source document of the web page in order to extract data. The parser, which is developed by Haines in [68] was the main resource for building the DE application in the thesis. Figure 3.4 illustrates the pseudocode of the parser modified from the Haines Java codes.

This parser is tested to develop a simple application, “Stock Retrieval”, to extract structured data from a business web site. The application is developed based on the Haines’s application “Stock Tracker Application”. Only those classes which are used for the extraction process are selected. Several methods in many classes have been changed to be rendered compatible with the currently released version of the software used (e.g. Java and PostgreSQL 8.3[1]), as well as the structure of the chosen

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



MICROSOFT CORP (MSFT)

21.96
-0.36 (-1.61%)

OPEN:	21.06	MARKET CAP:	195,346,780,000	P/E:	0.00
HIGH:	22.06	52 WEEK HIGH:	37.50	F P/E:	
LOW:	20.95	52 WEEK LOW:	20.65	BETA:	0.8839
VOLUME:	153,919,648	AVG. VOLUME:	0	EPS:	

Insider trading data for MSFTBrowse MSFT SEC Edgar filings

INTERNATIONAL BUSINESS MACHS (IBM)

82.07
-2.28 (-2.70%)

OPEN:	79.27	MARKET CAP:	109,973,800,000	P/E:	0.00
HIGH:	84.31	52 WEEK HIGH:	130.93	F P/E:	
LOW:	78.82	52 WEEK LOW:	78.82	BETA:	0.903
VOLUME:	12,288,091	AVG. VOLUME:	0	EPS:	

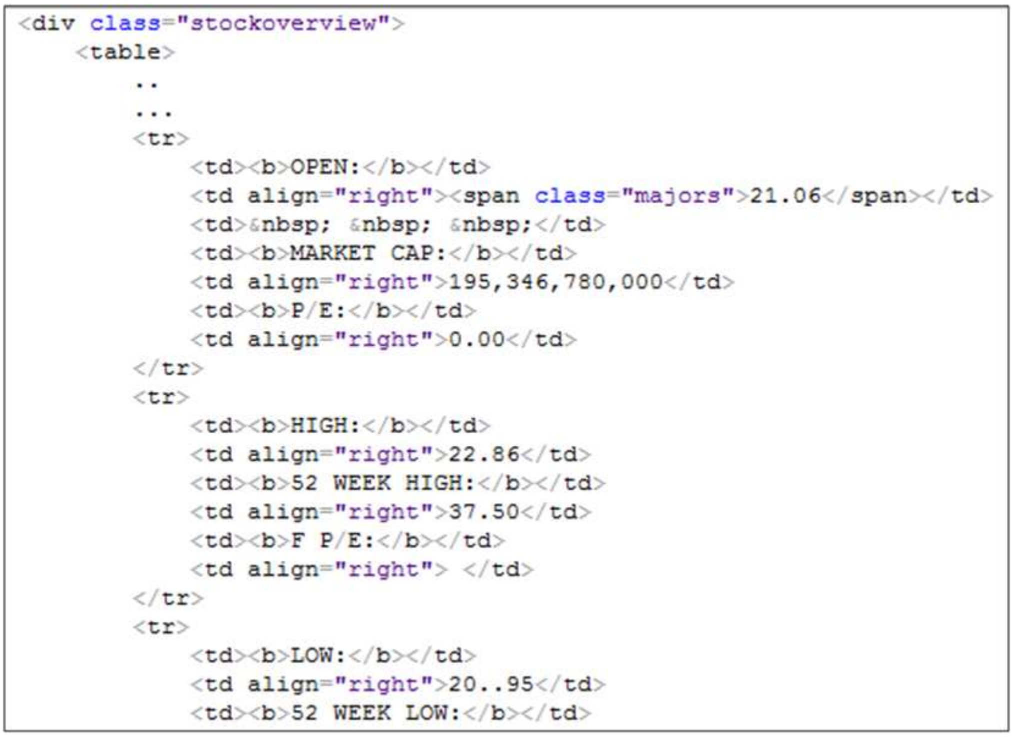
Insider trading data for IBMBrowse IBM SEC Edgar filings

Figure 3.5: Stock Quote Examples from PCQUOTE

web site.

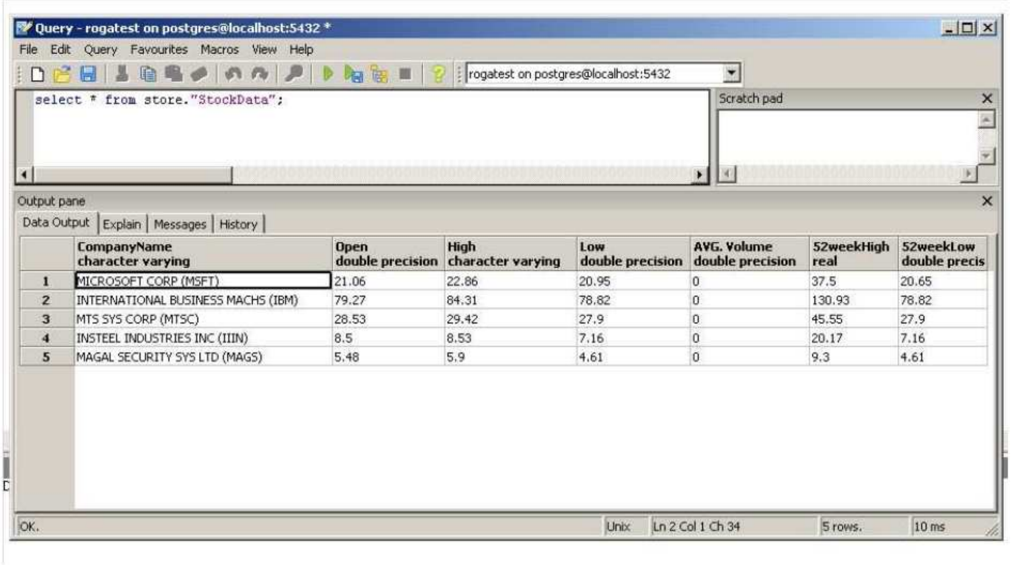
The Stock Retrieval application intends to extract stock data of an online stock market: in this case, `http://www.pcquote.com/stocks/`. The retrieved information is saved in a local repository for off-line analysis. The collected data set could be used for various purposes in business intelligence. Figure 3.5 shows examples of screen shots of the "PCQUOTE Stock Market" (taken in 2008).

Figure 3.6 shows the source web pages of the corresponding parts of the first table in Figure 3.5. Note that the data relevant for extraction from the source web page is found outside the HTML tags. In addition, the tags have no semantic to reflect or predict what the next data means. Thus, the application has to browse the source web page and splits its contents into tokens. If the token equals the open tag sign '<', then it will ignore all next tokens until it meets the close tag sign '>'. Data outside the tags will be extracted and saved in a vector to be sent to the local database. The the output and pseudocode of the Stock Retrieval application are as shown in Figure 3.7 and Figure 3.8 respectively.



```
<div class="stockoverview">
    <table>
        ..
        ...
        <tr>
            <td><b>OPEN:</b></td>
            <td align="right"><span class="majors">21.06</span></td>
            <td>&nbsp; &nbsp; &nbsp;</td>
            <td><b>MARKET CAP:</b></td>
            <td align="right">195,346,780,000</td>
            <td><b>P/E:</b></td>
            <td align="right">0.00</td>
        </tr>
        <tr>
            <td><b>HIGH:</b></td>
            <td align="right">22.86</td>
            <td><b>52 WEEK HIGH:</b></td>
            <td align="right">37.50</td>
            <td><b>F P/E:</b></td>
            <td align="right"> </td>
        </tr>
        <tr>
            <td><b>LOW:</b></td>
            <td align="right">20..95</td>
            <td><b>52 WEEK LOW:</b></td>
```

Figure 3.6: Part of the HTML Source Text of the PCQUOTE web page

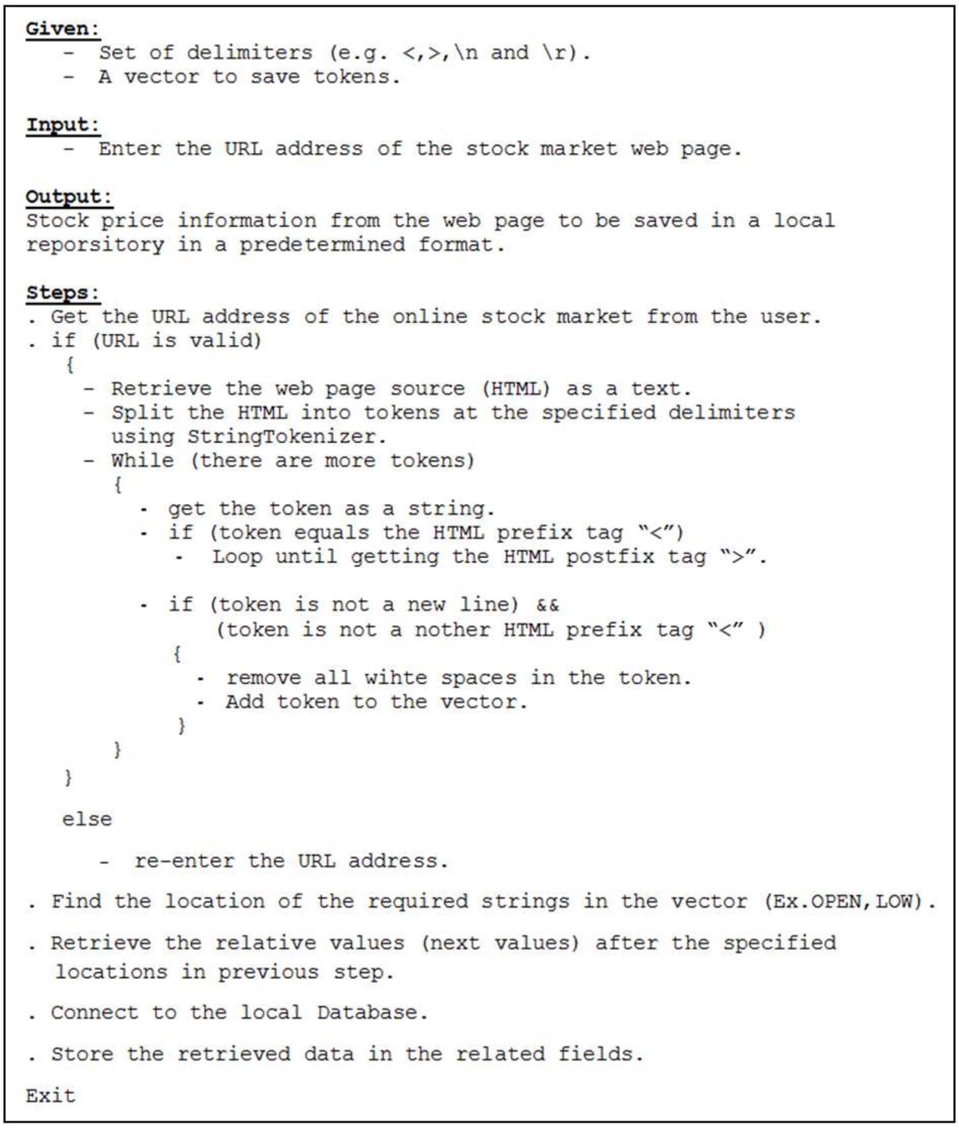


```
select * from store."StockData";
```

	CompanyName character varying	Open double precision	High character varying	Low double precision	AVG. Volume double precision	52weekHigh real	52weekLow double precis
1	MICROSOFT CORP (MSFT)	21.06	22.86	20.95	0	37.5	20.65
2	INTERNATIONAL BUSINESS MACHS (IBM)	79.27	84.31	78.82	0	130.93	78.82
3	MTS SYS CORP (MTSC)	28.53	29.42	27.9	0	45.55	27.9
4	INSTEEL INDUSTRIES INC (IIIN)	8.5	8.53	7.16	0	20.17	7.16
5	MAGAL SECURITY SYS LTD (MAGS)	5.48	5.9	4.61	0	9.3	4.61

OK. Unix Ln 2 Col 1 Ch 34 5 rows. 10 ms

Figure 3.7: Sample of the Content of Stock Retrieval Table in Local DB



```
Given:
   -  Set of delimiters (e.g. <,>,\n and \r).
   -  A vector to save tokens.

Input:
   -  Enter the URL address of the stock market web page.

Output:
Stock price information from the web page to be saved in a local
reporsitory in a predetermined format.

Steps:
. Get the URL address of the online stock market from the user.
. if (URL is valid)
   {
     - Retrieve the web page source (HTML) as a text.
     - Split the HTML into tokens at the specified delimiters
       using StringTokenizer.
     - While (there are more tokens)
       {
         · get the token as a string.
         · if (token equals the HTML prefix tag "<")
            ·  Loop until getting the HTML postfix tag ">".

         · if (token is not a new line) &&
                (token is not a nother HTML prefix tag "<" )
            {
              ·  remove all wihte spaces in the token.
              ·  Add token to the vector.
            }
       }
   }
   else
      -  re-enter the URL address.
. Find the location of the required strings in the vector (Ex.OPEN,LOW).
. Retrieve the relative values (next values) after the specified
  locations in previous step.
. Connect to the local Database.
. Store the retrieved data in the related fields.
Exit
```

Figure 3.8: Pseudocode of Stock Retrieval Application

3.3 Data extraction from OSN

OSNs are a special part of the web where several interesting phenomena take place. The information in an OSN can provide a good collection of resources, for researchers from different disciplines such as psychology, sociology and computer science to study. The changes in the users' profiles affect their networks' behaviour and actions and lead to alterations in the pattern analysis. In terms of human aspects, online social network (OSN) providers cannot provide users data to any third parties because of the privacy regulations. Thus, more attention is required on how information will be collected from OSN websites to cope with the rapid changes [47, 128].

Most of the early research in DE from OSNs either does not mention how the data has been extracted or relies on non automated approaches to extract information, e.g. conducting surveys and interviews, then analyzing the results. Table 3.2 summarizes some of this research [16, 56, 66, 124]. Although the research presents some interesting findings, for instance in studying the disclosure and privacy behaviours of students in colleges and universities, the samples upon which the findings and results are based are too small (a few hundreds in the best cases). In addition, the samples were recruited in an ad-hoc manner instead of via random sampling. Thus, the sample may not reflect the real behaviour of the targeted phenomenon, especially if participants did not tell the truth.

Table 3.2: Non Automated Data Extraction from OSN

Research study	Advantages & Disadvantages
[56] 2007 The research compares the trust and privacy concerns of users of two OSNs as well as the desire to share information and form new relationships. **Methods** Online survey of users of two OSNs: Facebook and Myspace.	**Advantages** The research is qualitative in comparing the users behaviours in two most popular OSNs rather than one, as most research has done. **Disadvantages** Small sample (total of 117 participants: 69 Facebook users and 48 MySpace users).
[66] 2007 The research investigate s how students unknowingly played an important role in posing themselves and their campus networks to malicious attacks. **Methods** • Interview security experts. • Conduct surveys with university students.	**Advantages** Presents steps to reduce the effectiveness of an attack, e.g. instituting a combination of policies and controls, and increase students security awareness of the risks of using OSN. **Disadvantages** Small sample (41students)
[124] 2007 The research examines qualitatively the disclosure and privacy behaviours and attitudes of the students. **Methods** • Interview university students. • Survey of university students.	**Advantages** The research points to the impact of the complexity and ambiguity of the Facebook interface in utilizing the appropriate privacy. **Disadvantages** Small sample (12 active Facebook users).

Table 3.2: (continued)

Research study	Advantages & Disadvantages
[16] 2006 The research explores the patterns and motivations of students to disclose information on Facebook. The study concludes that the widespread public attention on the privacy risks of Facebook affects its users. **Methods** • Conduct surveys with college students. • Using Perl scripts to download and parse Facebook HTML pages (when Facebook was limited to students).	**Advantages** The research methodology is explained very well and uses two different approaches to gathering data (manual and automated). In addition, the results are based on a sample which is filtered precisely. **Disadvantages** Small sample of survey (294 out of 506 actually completed the whole survey) compared with 7,000 profiles parsed using Perl scripts.
[77] 2008 The study finds out empirically the type of posted information on Myspace OSN by youth. **Methods** • Use generators to select Myspace profiles randomly. • Extract data from profiles manually through filling in data collection forms.	**Advantages** The study provides useful recommendations based on statistical analysis of profiles contents to youth and parents in order to increase the awareness of disclosure their personal information publicly. **Disadvantages** Too exhaustive and slow work to extract and analyze an average of 16 profiles per day by each staff, due to 8406 public profiles out of 9282 are randomly selected within period (June- August 2006) by staff of 6 (2 author and 4 assistants).

Moreover, these approaches are time consuming and inconvenient nowadays, especially with the rapid changes in the amount of information uploaded to OSNs on a daily basis. Recent research has relied more on automated approaches such as crawlers, which are used in the majority of research developed.

Table 3.3 presents some of these studies. Note that the criticism of literature is based on techniques used for data extraction. The advantages and disadvantages are not limited to what are mentioned in the table, and some of the cells regarding disadvantages are empty due to the fact that the disadvantages are not related to the field of data extraction.

Table 3.3: Automated Data Extraction from OSN

Research study	**Advantages & Disadvantages**
[128] 2009 Extract the interaction between more than 60,000 Facebook users and study the evolution of the activity of a given user profile and friends of their friends over 2 years. They found that the majority of interactions from over than 800,000 logged in interactions are generated by a minority of user pairs. **Methods** • Build a crawler. • Download HTML documents. • Use BFS fashion to select the next profile.	**Advantages** The study retrieves the history of users profiles through their wall posts and analyzes the results based on two groups: • Low level interaction: pair of friends who take more than 1 month to start wall posts. • High level interaction: those who initiate conversations immediately after becoming friends. **Disadvantages** • The data is collected just twice within 2 years. • Different data is collected each time: 1- just friendship links are collected. 2- the profile walls of users who are crawled in the first collection.

Table 3.3: (continued)

Research study	Advantages & Disadvantages
[35] 2009 The study focuses on the technical aspects of how data will be extracted from Facebook OSN. It examines the difficulties in collecting profile and graph information as well as describing various approaches to extraction of data by third parties such as public listing, false profiles, profile phishing, malicious applications and Facebook query language (FQL). Also, the study evaluates each of above techniques to measure their effectiveness against profiles' privacy settings. They conclude that the existing privacy protection systems in OSNs are not consistent with users' expectations. **Methods** • Build a crawler. • Use Facebook query language (FQL) to collect data.	**Advantages** Extract information from Facebook profiles using 3 algorithms in order to study the privacy issues of OSN users: • Public listing: to crawl public profiles. • False profiles: to crawl searchable profiles via creating false profiles. • Profile compromise and phishing: to crawl random or specific accounts through malicious applications and phishing attacks. **Disadvantages** The crawler script collects only profile IDs and friendship links.

Table 3.3: (continued)

Research study	Advantages & Disadvantages
[45] 2008 The research extracts data using two approaches: • Random sampling: since the Myspace profiles IDs are sequentially numbered, the sample is chosen by generating random IDs. • Relationship based sampling: provided a random set of IDs, the sample is chosen from the friends lists of these IDs to have a connected structure of network. The extracted data from both samples is analyzed and compared in order to understand who is using the OSNs and how these networks have been used. **Methods** Build 2 Myspace-specific crawlers based on two modules: • Perl's LWP::UserAgent. • HTML::Parser.	**Advantages** Extract a large-scale study of Myspace OSN (over 1.9 million profiles). Analyze the extracted data from 3 aspects: • Sociability: how users participate in OSN. • Demographics: how users describe themselves. • Language model: how users share their feelings and interests.

Table 3.3: (continued)

Research study	Advantages & Disadvantages
[52] 2004 The study uses email contents as inputs to extract a large network containing the user's friends of friends of friends, and to build the system as follows: • Extract the names and email addresses from the email headers of the user. • Use a set of string matching rules to find out the multiple representation of the same person. • Find the home page of each person based on his name and interested domain . • Employ some probabilistic information extraction models to find all information about this person and all the people found on his webpage and build a social network. • Create a list of keywords for each person. • Analyze and cluster the social network by graph partitioning algorithms. **Methods** • Use a machine learning approach to extract contact information from web pages. • Use Conditional Random Field (CRF) to build a probabilistic model of these fields.	**Advantages** The system can • Find experts in large communities. • Automatically recommend experts on particular topics. • Build and analyze the social network graph to find a path between users and cluster their attributes.

Table 3.3: (continued)

Research study	Advantages & Disadvantages
[102] 2007 The study aims to analyze the structural properties of OSN. The data is collected by crawling 4 OSNs using automated scripts on a cluster of 58 machines. **Methods** • Build a crawler which relies on API and screen scraping the HTML pages. • Use BFS algorithm.	**Advantages** The authors claim that their study is the first which extracts data from multiple OSNs at scale. • Extract data from 4 OSN: Flicker, YouTube, LiveJournal and Orkut. • Extract large scale data set (over 11.3 million users and 328 million links). **Disadvantages** The data collected by the crawlers is limited to the URL of user profiles and their friends for the 4 OSNs.

Unlike Mislove et al. in [102], who simply retrieves profiles' URLs for users and their friends, the research in this thesis concerns extraction of all accessible information on the profile. However, this thesis uses API as well as screen-scraping to parse the HTML pages. Viswanath et al. in [128] are concerned with extracting historical data over time, as in the aim of this thesis. However, their approach limits the historical data extracted to the wall posts. Moreover, despite the fact that data is collected over 2 years, the collected data could be categorised into two types, and each is collected just once. The first type is the profile IDs and their friendships links, while the second is the wall posts of these profiles.

Bonneau et al. [35] present a unique study in discussion of DE from OSNs from a technical point of view. However, they limit the data extracted to only profile IDs and their friendship links, and this only once. In contrast, this thesis focuses on extracting all information which can be crawled.

Caverlee et al. in [45] present a very interesting approach to choosing the seed IDs of the sample. The seed ID is the first profile ID to start a crawling process. The samples are used to compare the results based on connected and disconnected sub networks.

The approaches adopted for this thesis, as will be described in Chapters 4, 6 and 7, are similar to [128, 45, 49] in terms of using a Breadth First Search (BFS) algorithm to travel across the OSN, since it is optimal, easy to implement and more representative to the friendship relations. Alike [45], profiles that have the structures associated with bands, comedians or musicians are rejected because they do not represent friendships between individuals. In fact, this is required by a member of the co-authors in the papers published in [20, 9, 8] to calculate the vulnerability measurement of individuals in OSNs.

3.4 Seeking for Information using MAS

Despite the power of online information seeking tools in locating matching terms and phrases on the web, these tools are considered passive systems as claimed in [104]. MAS transforms these passive information seeking tools into active personal user assistants through merging the agent technology with effective information seeking techniques in order to improve the performance of information seeking tools. Table 3.4 illustrates some of the previous works attempting to seek information using MAS.

This thesis uses an html parser with agent as in [123], and JADE, as [101] does not mention how the data has been collected by agents. However, their approach, which uses the JXTA agent communication method to communicate through firewalls and network address translators, could be suggested as future work for the systems developed in this thesis.

Table 3.4: Seeking Information using MAS

Research study	Advantages & Disadvantages
[101] 2007 The study proposes architecture for MAS to retrieve information using the fuzzy logic concept. When the user submits the query to the system, an agent called a Broker Agent (BA) passes the query to several Search Agents (SA): each looks for information in different web database servers to return an ordered list of results to a Response Agent (RA). RA must rank all results returned from SAs and apply some methods based on associating each server with a fuzzy set of key terms most representing the topics. The approach enables developers to build and deploy peer to peer applications to manage document retrieval from multiple sources. **Methods** • Using fuzzy logic based result fusion mechanism. • Using JADE framework to implement MAS.	**Advantages** Use JXTA agent communication method to communicate through firewalls. **Disadvantages** Does not mention how SAs collect data.

Table 3.4: (continued)

Research study	**Advantages & Disadvantages**
[123] 2001 The study presents three crawler agents to test their performance in retrieving biomedical information about diseases when information about genes is given. The first two agents are single-agent algorithms based on Best-First traversals (BFSN) where N =1 and N=256. BFS1 is the more studied algorithm and BFS256 is a more explorative agent.The third agent is called the InfoSpider agent. It is a multi agent approach based on an evolving population of learning agents. Starting from a seed set, each agent visits up to 1000 pages per topic, chopping pages larger than 100 KB. A maximum buffer of 256 links is allowed for each agent to be tracked. The agent must decide to replace some of these links if the buffer is full. The experiment shows that the InfoSoft agent performs best in the static similarity metric. **Methods** • Using MAS implemented in Perl. • Using simple HTML parser.	**Advantages** The authors claim that their study was the first in using an agent to crawl web pages for biomedical information. **Disadvantages** • Although the methodology used to select seeds ensure the existence of the page and the links, some of the seeds found do not exist or have no links. • Some of the retrieved pages are not in English, which causes some mismatch. • Some of the topics were eliminated regarding the cryptic nature of the gene names, gene product names and gene symbols which leads to results in different areas. E.g. BACH1 is a symbol of gene product but leads to the composer Bach in the music world instead of the breast cancer field.

Table 3.4: (continued)

Research study	**Advantages & Disadvantages**
[104]2004 The study aims to minimize the total completion time to traverse routes for retrieving information using mobile agents. The user initiates multiple mobile agents concurrently. Each agent visits the providers which host the information and retrieves a subset of required information to be returned to the user. **Methods** • Using intelligent agent. • Using evolutionary computing as in search optimization and learning algorithms. • Using fuzzy logic.	**Advantages** The author claims that the study is unique (at the time of publication) in the way in which the intelligent agents are directed and in combining the computational intelligence techniques with intelligent agents to improve filtering and retrieval of information.

Table 3.4: (continued)

Research study	**Advantages & Disadvantages**
[125] 2005 The study proposes architecture for MAS to retrieve information using the fuzzy logic concept. When the user submits the query to the system, an agent called a Broker Agent (BA) passes the query to several Search Agents (SA): each looks for information in different web database servers to return an ordered list of results to a Response Agent (RA). RA must rank all results returned from SAs and apply some methods based on associating each server with a fuzzy set of key terms most representing the topics. The approach enables developers to build and deploy peer to peer applications to manage document retrieval from multiple sources. **Methods** Applying 3 heuristic methods: • Saving. • Insertion. • Serial Reverse-Constructed Route(SR-CR).	

3.5 Data extraction from OSN using MAS

To the best of the authors' knowledge, existing research in extracting data from OSNs using MAS technology is limited, due to the fact that OSNs have become popular only recently, within the last decade, although the concept of OSN has been known since the 1997 as mentioned in Section 2.4.2.

The only research that is found in this area is that of Chau et al. in [47], who extend the parallel crawler designed by Cho et al. in [49]. While Cho et al. work on the static assignment architecture, where each crawler is assigned to a part of the web (in general) to retrieve information and coordinate with each other without a central coordinator, Chau et al. work on dynamic assignment architecture to crawl the ebay[1] network using a central coordinator (Master Agent) to control all crawlers.

Chau et al. built a parallel crawler for crawling ebay users' profiles through the list of IDs of those users who left feedback on the main user profile. The crawler system is designed for NetProbe: a fast and scalable system for fraud detection (see [110]).

Moreover, situations which require a continuous observation of the OSN user's profile in order to track the changes that could help in understanding the structure of OSNs and its effects on different disciplines have not been addressed widely.

3.5.1 Data Extraction using API:

In this thesis, the focus is on using API programmatically to enable software developers to extract data from OSN profiles. To the best of the authors' knowledge, little research has reported on using API to extract data from an OSN. For example, Mislove et al. in [102] collected a data set that includes over 11.3 million users and

[1] http://www.ebay.com

328 million links from Flickr[1], YouTube[2], LiveJournal[3] and Orkut[4] through integrating their crawlers with the API supported by OSN providers. The retrieved data is used to study and analyze the structure of those OSNs. The results show that OSNs have a far higher fraction of symmetric links as well as higher levels of local clustering. The researchers stated how these properties could affect designing OSN algorithms and applications.

Another study was undertaken by [22], who used API to look for the term "university" in the name or description of Flickr groups. The aim was set to investigate if the Flickr community could benefit from the tags that are used within the university image groups. They retrieved a sample of 250 random images together with image tags, uploaded by those groups. They found that the uploaders of the images pay attention to assigning multiple tags (at least four) on each image, which is useful for image retrieval purposes.

The work presented in this thesis is similar to [47] in terms of using BFS algorithm to crawl the OSN profiles as well as using MAS to extract data in parallel. However, the MasterAgent must assign which agent is to crawl the next OSN user in the queue. Moreover, the grabber agents are designed to monitor the profiles of OSN users to build a history for each user rather than visiting each user only once. Note that details related to how the crawler is built in [47] have not been presented.

Catanese et al. in [44] extracted data in August 2010 when it was allowed for third parties to access the Facebook Friends of Friends (FOF) profiles. They used a queue to list the pending users who are seen but not visited yet using Breadth First Fashion. A central coordinator and data master is created to control the queue in order to ensure that no redundancy occurs in visiting the ebay users. Parallel

[1] http://www.flickr.com
[2] http://www.youtube.com
[3] http://www.livejournal.com
[4] http://www.orkut.com

crawler agents are distributed to crawl the ebay profiles. Each crawler agent sends a request to the master in order to get the next user from the queue. The crawler agents use multiple threads for crawling then return the extracted information to the master. Within 3 weeks, they had visited more than 11 million profiles, and 66,130 profiles had been visited completely (i.e. with all users in the feedback list having been crawled).

Although these parallel crawlers were termed agents, the authors did not pay any attention to considering their crawler system as a MAS.

3.6 Formalizing MAS

In Section 2.6.2, the importance of using formal specification in describing the system before implementation was explained. Moreover, Hilaire et al. in [76] define two roles for the specification process to fulfil. Firstly, it should provide the underlying rationale for the developing system. Secondly, it should guide in the design, implementation and verification phases of the system. MAS used to be considered a complex system that needed to be formalized [132, 39]. Thus, various research approaches have been evolved in the last two decades to face the problem of developing MAS. Figure 3.9 which is modified from [39] presents some of the formalisms used in formalizing MAS.

Boucherit et al. in [39] classified research on formalizing agents into two major approaches:

- Behavioural approach.
- Logic approach.

This section highlights some of these languages and methods.

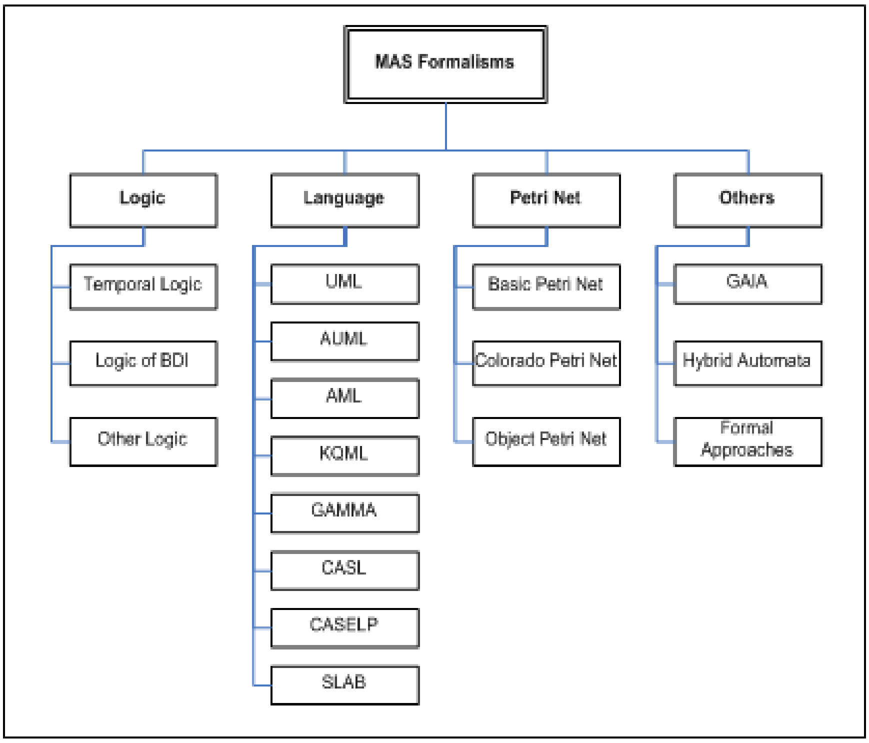


Figure 3.9: Formalisms Used for MAS Formalization Modified from [39]

Logic for Contract Representation (LCR)

The Logic for Contract Representation (LCR) language is developed by Dignum et al. in [54] as a very expressive logic to describe the interaction in MAS. LCR is based on branching-time logic: i.e. it uses a tree-type branching structure where the nodes represent the states and the arcs represent the events.

According to the possibilities that the agent has at each moment, states are linked to obtain courses of events in order to obtain what is called a path. LCR allows checking of whether the agents follow the required interaction patterns as well as whether the agent preserves the desired social states. One of the limitations of LRC is that it concentrates only on the logical representation of contracts while the interaction structure as a whole has not been investigated.

Specification Language for Agent-Based Systems (SLAB)

SLAB stands for Specification Language for Agent-Based Systems and is presented by Zhu in [134, 132, 133]. It focuses on behaviours of agents in its environment. Specification of the MAS in SLAB consists of the following:

System ::= Agent-description — caste-description

SLAB produces the concept of "Caste" which means a set of agents in a MAS that have common capabilities and behaviours to perform a certain task. Caste is a natural evolution of a class concept in object-orientation. Caste, as shown in Figure 3.10 (on the left) contains a description of the structure of its state and actions as well as a description of its behaviour and environment.

The second concept that SLAB presents is the "agent" (see Figure 3.10 on the right) which is similar to the object concept in object-orientation. An agent within castes differs from an object within a class in that the agent can be added or removed from the castes at run-time. Every agent in SLAB should be an instance of a caste. Consequently, agents inherit all a castes structural behaviour and environment descriptions. SLAB sets a default caste called AGENT if no caste name is given in the agent specification [132, 134]. Although SLAB is a powerful formal specification language [39], it is not used widely. Most research found in literature which uses SLAB in specifying the MAS refers to its founder as an author, e.g . [98].

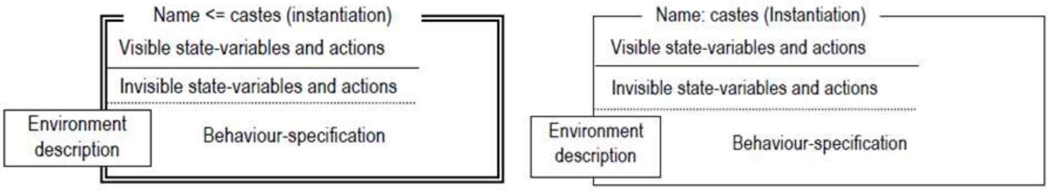

Name <= castes (instantiation)
Visible state-variables and actions
Invisible state-variables and actions
Environment description
Behaviour-specification
Name: castes (Instantiation)
Visible state-variables and actions
Invisible state-variables and actions
Environment description
Behaviour-specification

Figure 3.10: Graphical Description of Castes on the Left and Agents on the Right as Presented by [132]

Z Notation

Unlike all other languages, the formal specification language Z has been made an ISO standard for formalizing software systems. It is used widely in academia as well as in large projects in industry [81, 71, 40]. This attracts many researchers to adopt it in developing complex systems, e.g. MAS models. For example, Niazi and Hussain in [106] developed a simulation model for wireless sensor networks for monitoring Complex Adaptive Environment. They chose Z to present FABS, a novel Formal framework for Agent-Based Simulation [93]. However, Iglesias et al. argue that since Z does not have a notion of time, it would be less well suited to specifying agent interactions [80].

3.7 Chapter Summary

The chapter presented some of previous studies which are related to different approaches to extract data from the web in general and OSN in particular. Also the chapter explored research using MAS technology in seeking information from the web as well as from OSN. Additionally, some of the formal specification languages which are used to formally specify MAS are illustrated.

How the thesis gets advantages from these studies are described after each sub section as follows: in terms of the technique used for this thesis, the approach adopted is under HTML-aware tools, which is presented in [87], and as more likely to be under wrapper induction regarding classification in [21]. It is similar to [24] in terms of relying on token analyzers to find the data of interest in the HTML web page.

Regarding DE from OSN, the approaches adopted for this thesis are similar to [128, 45, 49] in terms of using a BFS algorithm to travel across the OSN, since it is optimal, easy to implement and more representative of the friendship relations. Alike [45], profiles that have the structures associated with bands, comedians or musicians are rejected because they do not represent friendships between individuals.

In terms of using MAS, the thesis uses an HTML parser with agent as in [123], and JADE toolkit to build MAS as [101], although they do not mention how the data has been collected by agents.

Chapter 4

An Approach to Data Extraction from an Online Social Network

Preamble

With the increased use of online social networking sites, data extraction from social networking profiles is becoming a major tool for business. What makes social networking profile data different is its semi-structured format. The structure and the presentation of profile data change all the time. In OSN, there is a lack of research into automated data extraction from web pages. This chapter will highlight our approach which is based on automated retrieval of the profile's attributes and listing of top and all friends from MySpace, one of the early known OSN.

4.1 Introduction

The popularity of OSN sites has increased the amount of personal data which is publicly available. This data is distributed on the web as user profiles, which are mostly in unstructured or semi-structured format, change all the time; not just in their content but in their structure as well.

To benefit from the wealth of personal data accessible in OSNs, data must be extracted in the first instance. This requires new extraction methods to be developed. Most of the existing methods collect data from OSN manually or do not mention how the data has been extracted. There is very limited research associated with automated extraction methods from OSN sites, as described in Chapter 3.

This chapter presents the thesis's first contribution to extract data from deep web, particularly from OSNs automatically. This is due to the fact that data extracted is not reachable by crawlers of search engines. The Conceptual overview of the proposed approach is described in detail in Section 4.2. The implemented application is presented in Section 4.3, and Section 4.4 discusses the experimental results. Section 4.5 describes the validation while Section 4.6 highlights the approach limitations. Finally, Section 4.7 summarizes the chapter.

Note that "user profile", "profile identity" and "URL address" are used interchangeably in the thesis. Also, if ”parser” is mentioned in experiments, it means parser with string tokenizing as a part of crawler.

4.2 Conceptual Overview of Data Extraction from OSN Web Pages

This section presents the algorithm developed to extract data from OSN web pages automatically. It concentrates on how to extract the displayed personal details of OSN profiles such as name, age, gender, location, etc. as well as the list of friends and their information through the following steps:

4.2.1 Specifying the Domain of Extraction

Although there are several sites for OSN, Myspace has been chosen as the domain of extraction for two reasons. Firstly, at the time of the extraction (2009, 2010), it was the largest OSN and the most visited web destination according to Caverlee and Web in [45] as well as shown in Chapter 2. Secondly and most importantly, Myspace is open, i.e. it allows a rich source of data to be derived from profiles without the need for membership.

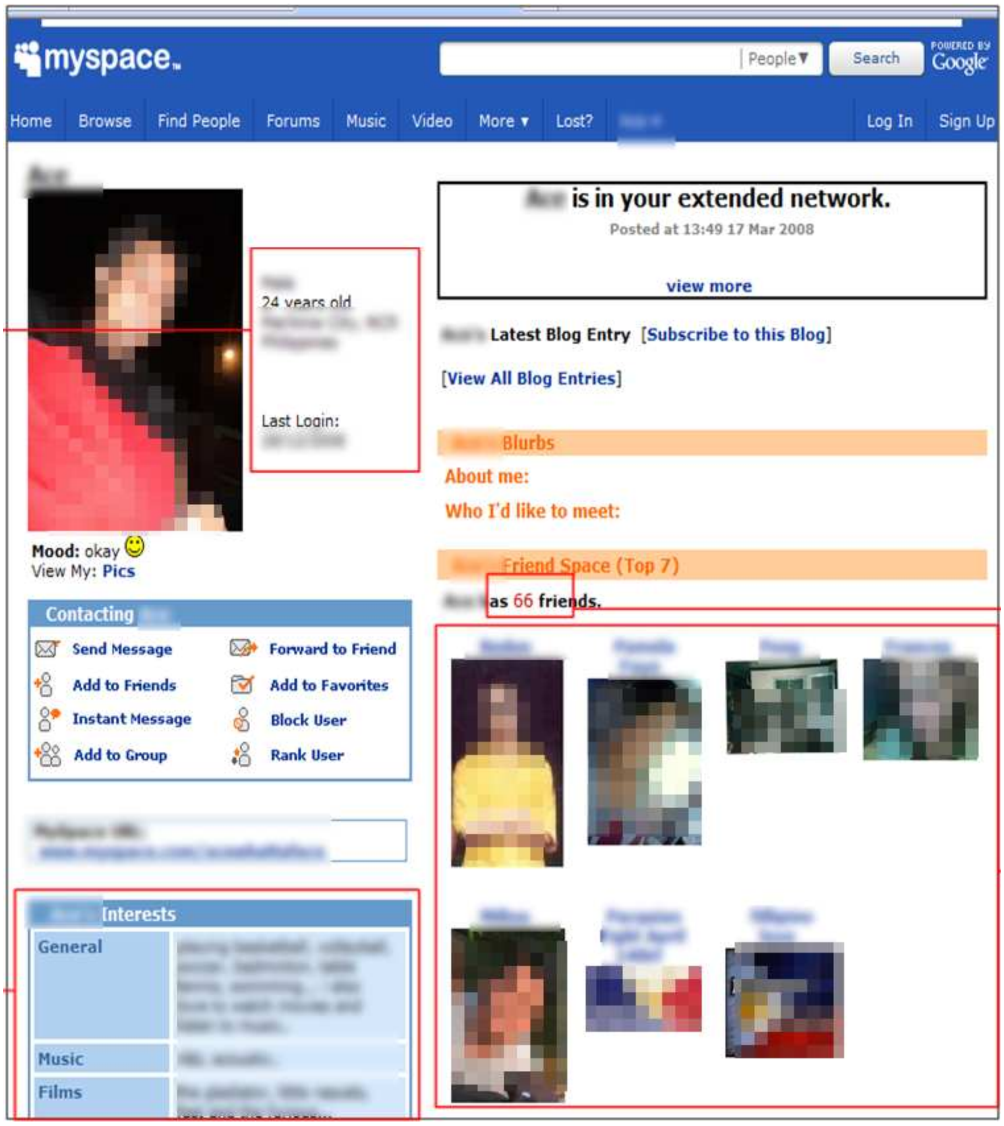


Figure 4.1: An Example of a Default Myspace User Profile

Figure 4.1 shows the default webpage of a Myspace OSN profile, where the basic information is displayed on the top left next to the profile picture, the detailed information in a blue table on the bottom left, and the list of top friends is on the bottom right. The list of all friends is obtained by clicking on the hyperlink (number in red).

However, Myspace allows the profiles' owners to customize their webpage to display the most interesting information. Figure 4.2 shows the way in which the

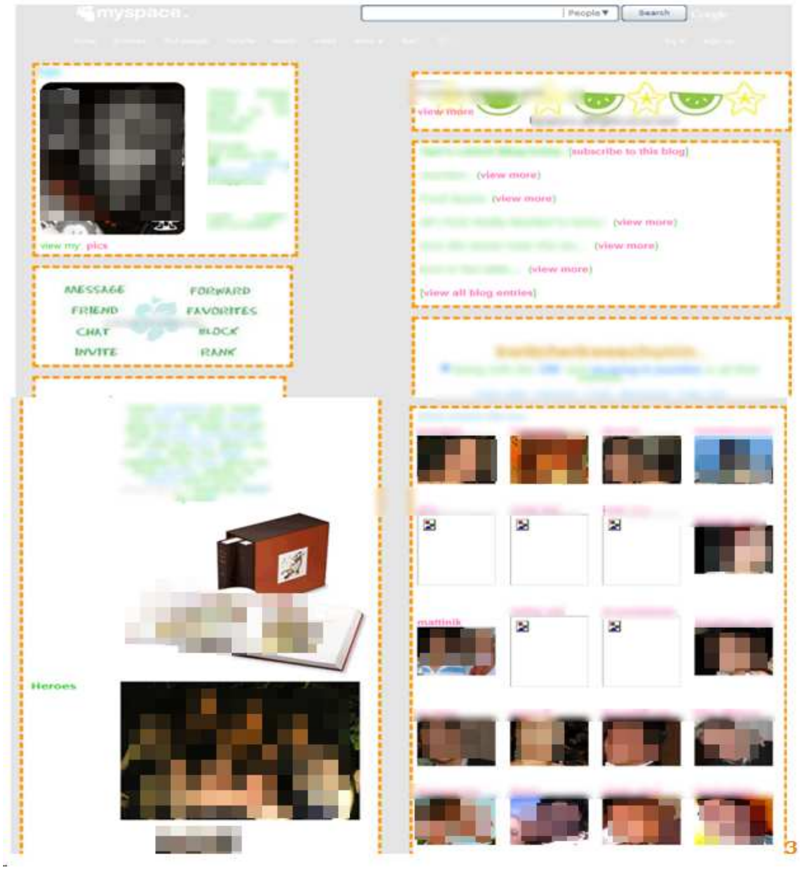


Figure 4.2: An Example of a Customized Myspace User Profile

page has been customized and the displayed information such as the list of top friends has been altered.

4.2.2 Contribution to Automated Data Extraction

In order to apply automatical extraction of data from OSN profiles, the selected approach relies on building a crawler: a popular method to retrieve data from a web page, as shown in Chapter 2. The crawler is a modified form of “The stock tracker application” which is presented by Haines in [68]. Haines’ application was designed to extract data from the online stock market, where the source of data is structured. However, building a crawler for parsing OSN profiles is more challenging for many reasons, including:

- OSN web pages are mostly in unstructured or semi-structured format, which is more difficult to deal with than a structured format. They contain a wide variety of data formats e.g. text, photos, videos, links etc.
- OSN web pages can be customized by users. Customization adds various effects to an already dynamic web page.

Figure 4.3 shows the approach developed to extract data from an OSN automatically. The details of the approach are as follows:

1. ***Data Pre-processing:***

As discussed previously in Section 2.3.3, building a web crawler depends mainly on the parser of the web page. This requires downloading of the source document (HTML) of Myspace profiles and analysis of the document.

Since Java is selected to implement the application, it would be simple to download the HTML of Myspace pages from the web through three steps:

- Create a `java.net.URL` object of the address of the first Myspace profile (seed node).
- Open a connection to that URL through the `OpenStream ( )` method.
- Read the data from the URL through one of the stream reader methods e.g. `InputStreamReader` or `BufferedReader` which translate the bytes read into characters.

Once the complete Myspace profile web page is being read as text, i.e. the HTML source document of the profile has been obtained, this document should be analyzed to pinpoint the required information.

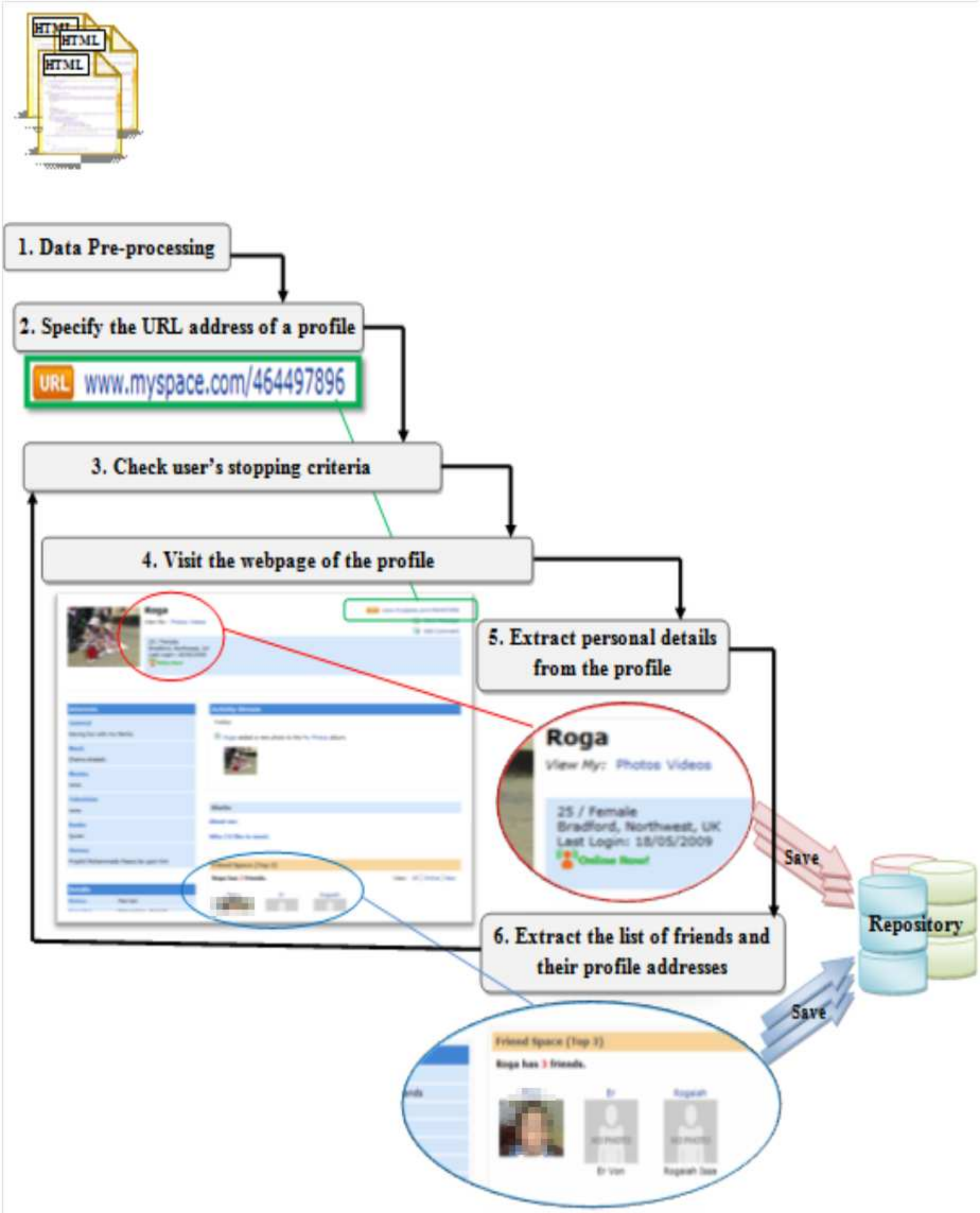

HTML
HTML
HTML
1. Data Pre-processing
2. Specify the URL address of a profile
URL www.myspace.com/464497896
3. Check user's stopping criteria
4. Visit the webpage of the profile
5. Extract personal details from the profile
Roga
View My: Photos Videos
25 / Female
Bradford, Northwest, UK
Last Login: 18/05/2009
Online Now!
Save
Repository
6. Extract the list of friends and their profile addresses
Save
Friend Space (Top 3)
Roga has 3 friends.

Figure 4.3: Automated Extraction Approach for Myspace Profile

It is well known that HTML documents contain tags ‘<’, ‘>’ to encapsulate all formatting information and commands for the web page. For example, the HTML raw text: <B>`my name is:` </B> denotes that the format command starts with the tag “<B>” and ends with the tag “</B>” and the text “`my name is:`” between the tags will appear on the browser web page in bold.

Thus, all information displayed on the web page is located between the tags. In other words, the parser must strip out all HTML tags to obtain the actual information from the OSN profiles, which is referred to as the “token”. From the resulting set of tokens, the location of the required tokens to be extracted and saved can be found.

Figure 4.4 illustrates the pseudocode of the proposed parser as modified from Haines' Stock Tracker Application [68].

2. ***Getting the URL Address***

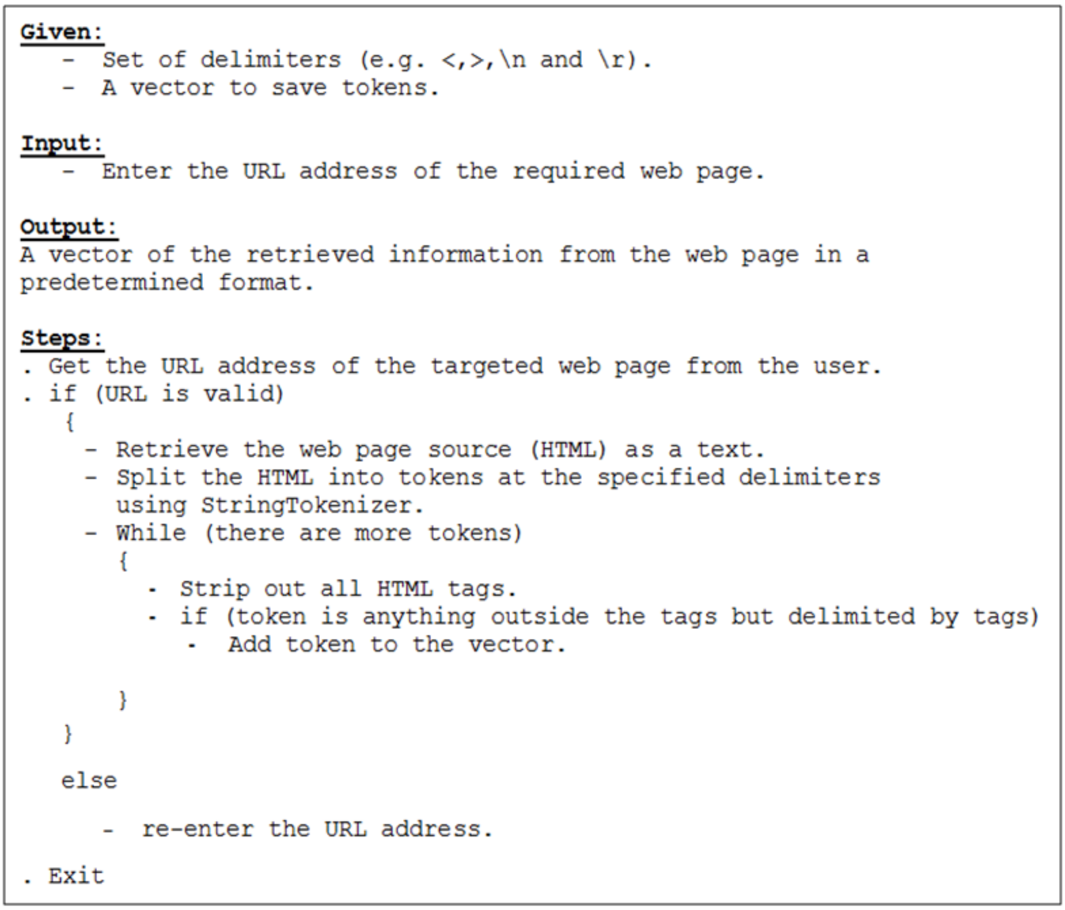
The approach begins by asking the user to specify two inputs. Firstly, it requests the seed URL of the Myspace profile. Seed URL means the first address of the Myspace profile to extract its information. This could be considered the central node of the resulting graph of the sub network. Each profile in all OSNs including Myspace has a unique URL address.

Once the user inserts the URL, the application should check it to insure that it is in the correct format. MySpace profile URLs should start with `http://www.myspace.com/` followed by the unique ID as shown in step 2 of Figure 4.3.

3. ***Checking the Stopping Criteria***

The second input that the application must collect from the user before starting the extraction process is the stopping criteria for quitting the application.

The user can specify whether the extraction should be stopped by:



```
Given:
    -  Set of delimiters (e.g. <,>,\n and \r).
    -  A vector to save tokens.

Input:
    -  Enter the URL address of the required web page.

Output:
A vector of the retrieved information from the web page in a
predetermined format.

Steps:
. Get the URL address of the targeted web page from the user.
. if (URL is valid)
   {
     - Retrieve the web page source (HTML) as a text.
     - Split the HTML into tokens at the specified delimiters
       using StringTokenizer.
     - While (there are more tokens)
       {
         · Strip out all HTML tags.
         · if (token is anything outside the tags but delimited by tags)
            ·  Add token to the vector.

       }
   }

   else

      -  re-enter the URL address.

. Exit
```

Figure 4.4: Pseudocode of HTML Parser Modified from Haines 2000

- The number of friends extracted.
- The level of iteration (e.g. a friend of a friend).

4. ***Visiting the Specified Profile***

Myspace classifies its profiles into different categories such as musician, bands, magazines, public or private profiles. The categories differ in their representation, attributes and structure, which affect the defining of the tokens in the parser.

Visiting the web page of the specified profile involves downloading the HTML document, as explained in data pre-processing stage, if the profile is not related to the musicians, bands or magazines categories. Otherwise, it will skip this profile and move on to the next profile in the queue of friends list. The queue will be explained later, in Section 4.3. The downloaded document will be read and translated into an array of characters.

5. ***Extracting and Saving Data from the Profile Webpage***

To extract the required data from a profile, the parsing method in [68] has been modified to enable it to work with the two different Myspace source documents as will be explained in Section 4.3.1. The extracted data includes all possible information about the profile users such as personal information (name, age, gender, photo link, etc.), detailed information, (smoker, drinker, etc.) and comments. If the information is specified by the user, it will be extracted. Otherwise, it will be replaced by null.

6. ***Extracting and Saving a List of Friends and their Profile Addresses***

There are two kinds of friends lists: a list of top friends, with whom it is assumed that the user might have a strong affiliation, and a list of all friends. Myspace will select by default, if the profile owner has not customized it, a subset of friends to be in the list of top friends, usually based upon the oldest friendships. Only top friends would be displayed in the main profile web page as shown in Figures 4.1 and 4.2. It was necessary to distinguish between these lists because they affect the choice of which profile should be the next for crawling. The resultant network will vary accordingly, and will influence data analysis and the relationship between network components.

For both kinds, a Breadth First Search (BFS) algorithm was used to select the next profile for extraction. The scenario below illustrates how BFS will be applied to the graph in Figure 4.5.

The graph represents a sub network of OSN where the vertices or nodes represent the profiles and the edges are the friendship relationships. The first node (seed) has two friends in its friends list which are nodes A and B. Node A has three friends which are: seed, C and D. Node B is a friend of three profiles: C, G and F.

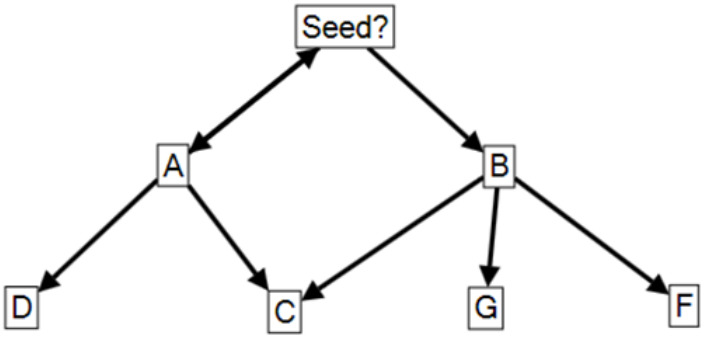


Figure 4.5: A Graph to Illustrate Breadth First Search

The BFS visits the nodes level by level. Thus, traversing the graph would be as follows:

1. Add the node seed, which is in level 0, into the front of the queue.

2. Obtain its friends list, nodes A and B, and add them to the rear of the queue.

Here the first level of iteration is completed.

3. Loop (while there are more friends in the level)
    (a) Look at the next node at the front of the queue (in this case node A)
    (b) Obtain its friends list: seed, C and D.
    (c) If the node does not exist in the queue (C and D), then add it to the rear of the queue. Otherwise, skip it (e,g, seed).
    (d) Repeat steps 'a', 'b' and 'c'.
4. By completing all nodes, the second level is ended.

The process will be repeated by obtaining the next URL in the front of the queue until the condition of the stopping criteria is met. The pseudocode in Figure 4.6 illustrates the extraction process.

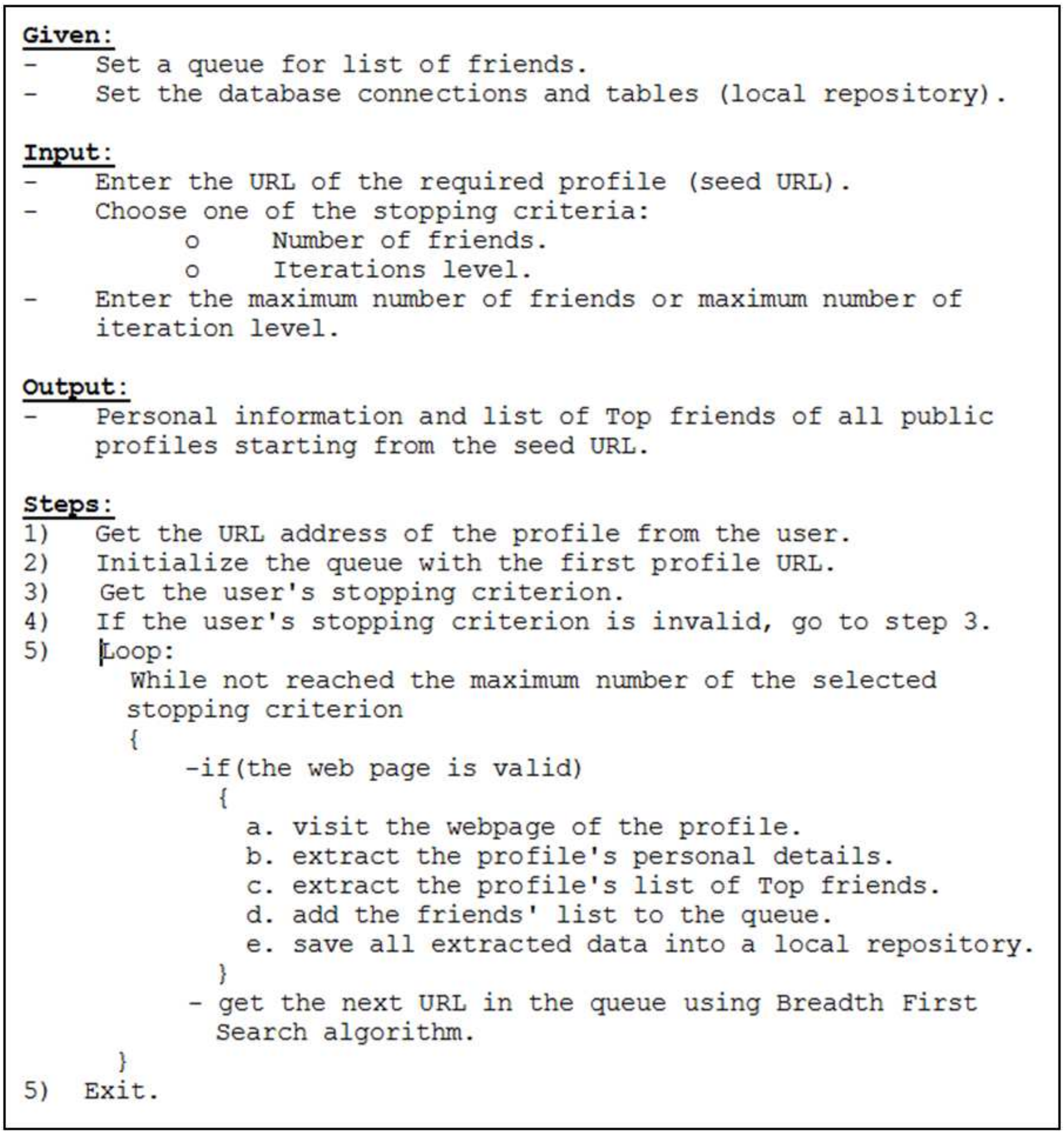
4.3 Implementation of Social Networking Extractor Application

An application called "Social Networking Extractor" is implemented. The application is built as a crawler that obtains the seed URL, extracts information on the web page and finds friends links to crawl their profiles as well. To clarify the developed application, the flowchart in Figure 4.7 details the steps of each stage in the extraction process.

The extraction process divided into three stages: Data pre-processing, Input initialization, and data collection and processing as detailed below:

4.3.1 Data Pre-processing

The Social Networking Extractor application is a sort of crawler that relies mainly on the parser. As stated in Chapter 3 and Section 4.2.2, the parser takes the



```
Given:
-    Set a queue for list of friends.
-    Set the database connections and tables (local repository).

Input:
-    Enter the URL of the required profile (seed URL).
-    Choose one of the stopping criteria:
           o     Number of friends.
           o     Iterations level.
-    Enter the maximum number of friends or maximum number of
     iteration level.

Output:
-    Personal information and list of Top friends of all public
     profiles starting from the seed URL.

Steps:
1)   Get the URL address of the profile from the user.
2)   Initialize the queue with the first profile URL.
3)   Get the user's stopping criterion.
4)   If the user's stopping criterion is invalid, go to step 3.
5)   Loop:
       While not reached the maximum number of the selected
       stopping criterion
       {
           -if(the web page is valid)
             {
               a. visit the webpage of the profile.
               b. extract the profile's personal details.
               c. extract the profile's list of Top friends.
               d. add the friends' list to the queue.
               e. save all extracted data into a local repository.
             }
           - get the next URL in the queue using Breadth First
             Search algorithm.
      }
5)  Exit.
```

Figure 4.6: Pseudocode of the Automated Data Extraction Approach

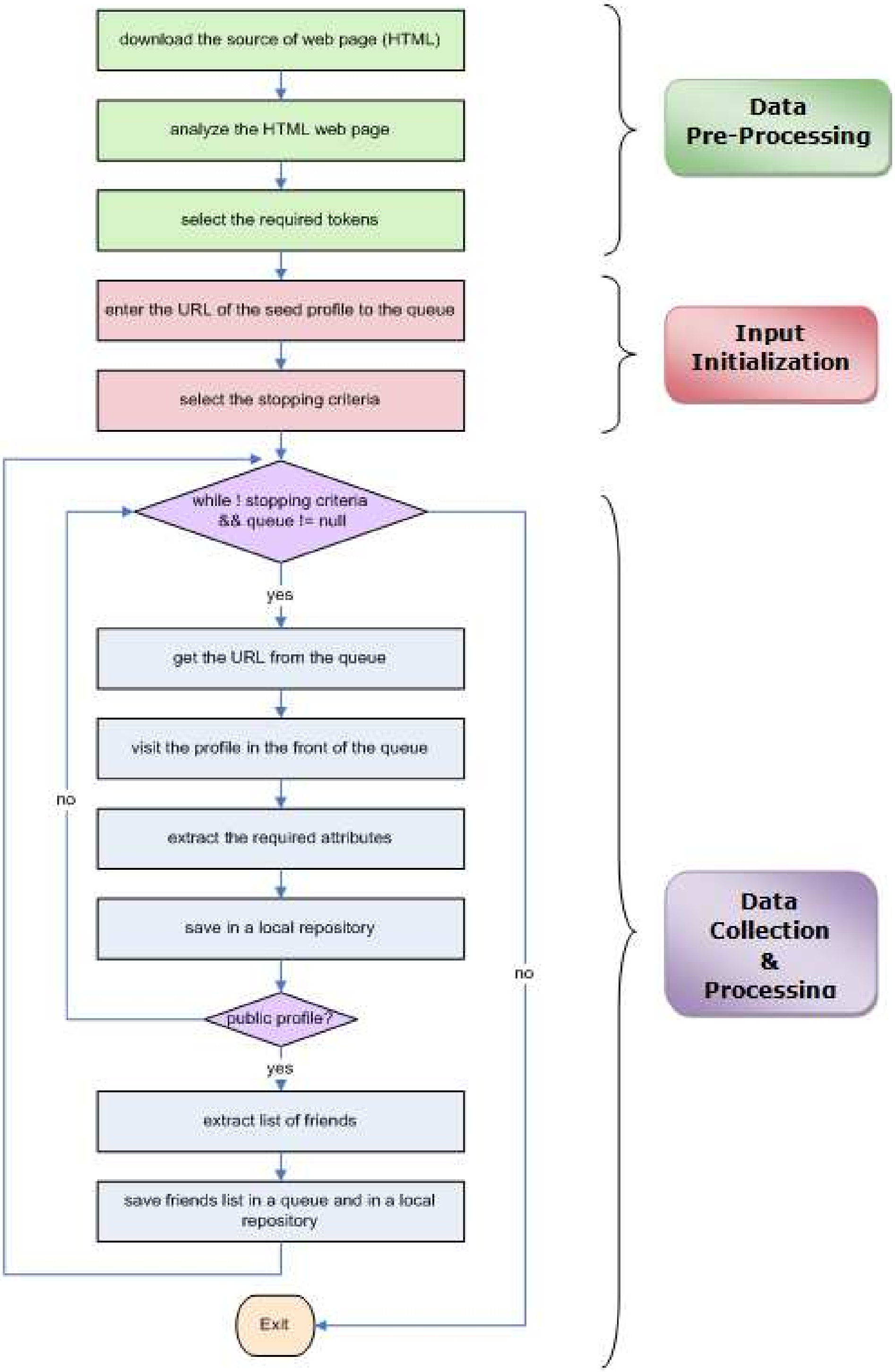
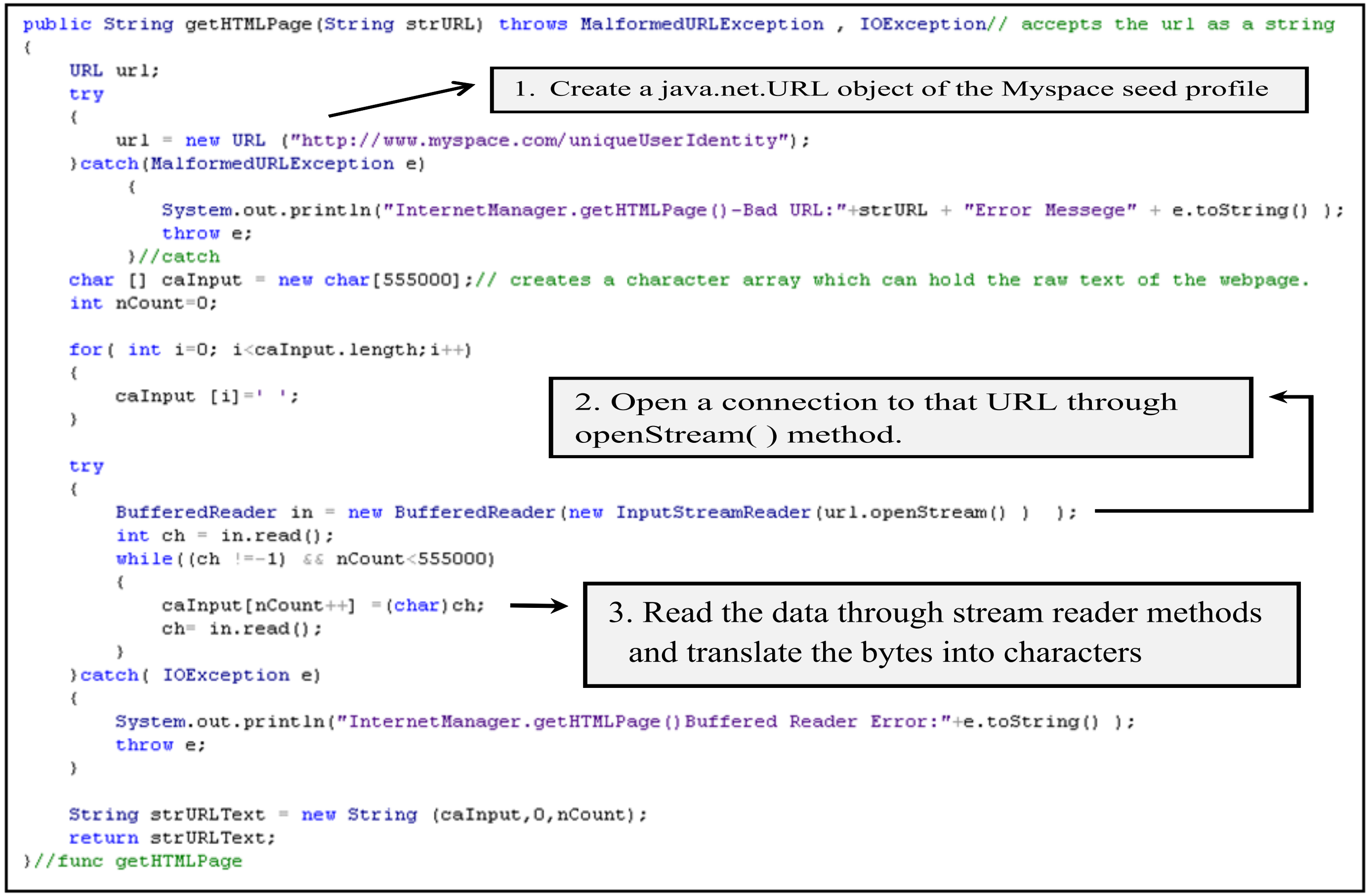

download the source of web page (HTML)
analyze the HTML web page
select the required tokens
enter the URL of the seed profile to the queue
select the stopping criteria
while ! stopping criteria && queue != null
yes
get the URL from the queue
visit the profile in the front of the queue
extract the required attributes
save in a local repository
public profile?
no
yes
extract list of friends
save friends list in a queue and in a local repository
no
Exit
Data Pre-Processing
Input Initialization
Data Collection & Processing

Figure 4.7: Flowchart of the Social Networking Extractor Application

source document (HTML) of Myspace profiles as a stream of Unicode characters to the tokenization stage where the tokens are classified. This requires downloading of the HTML document through three steps using the created Java class: `InternetManager` as shown in Figure 4.8.

The next step of the data pre-processing stage after reading the source profile's web page involves analysing it in order to find the values and locations of the profile attributes. Although MySpace allows external users to access the profile contents, the author created several MySpace profiles for two reasons. Firstly, to help in investigating all possible structures and attributes to develop the parser and database tables, especially as it was found that for most of the profile attributes, it is not compulsory to complete them, and this reflects on the resulting profiles.



```
public String getHTMLPage(String strURL) throws MalformedURLException , IOException// accepts the url as a string
{
    URL url;
    try
    {
        url = new URL ("http://www.myspace.com/uniqueUserIdentity");
    }catch(MalformedURLException e)
         {
            System.out.println("InternetManager.getHTMLPage()-Bad URL:"+strURL + "Error Messege" + e.toString() );
            throw e;
         }//catch
    char [] caInput = new char[555000];// creates a character array which can hold the raw text of the webpage.
    int nCount=0;

    for( int i=0; i<caInput.length;i++)
    {
        caInput [i]=' ';
    }

    try
    {
        BufferedReader in = new BufferedReader(new InputStreamReader(url.openStream() )  );
        int ch = in.read();
        while((ch !=-1) && nCount<555000)
        {
            caInput[nCount++] =(char)ch;
            ch= in.read();
        }
    }catch( IOException e)
    {
        System.out.println("InternetManager.getHTMLPage()Buffered Reader Error:"+e.toString() );
        throw e;
    }

    String strURLText = new String (caInput,0,nCount);
    return strURLText;
}//func getHTMLPage
```

1. Create a java.net.URL object of the Myspace seed profile

2. Open a connection to that URL through openStream( ) method.

3. Read the data through stream reader methods and translate the bytes into characters

Figure 4.8: Screenshot of InternetManager Java Class

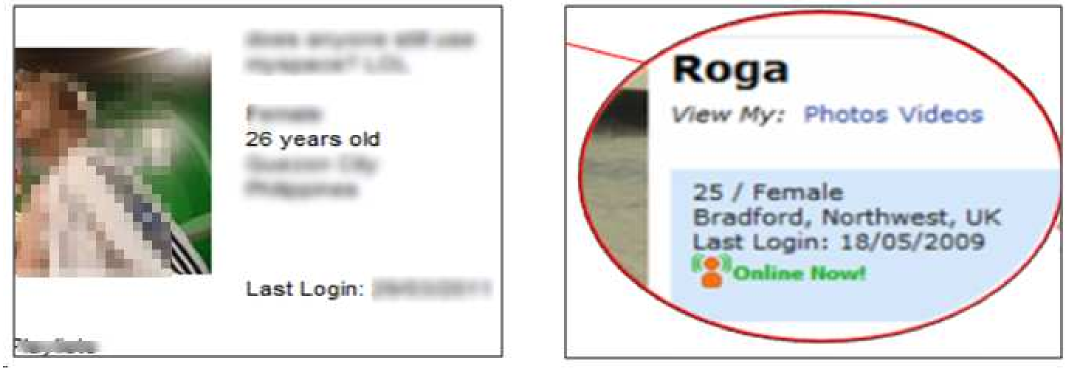
Basic information	**Details**	**Schools**
Display Name	Marital Status	Country
First Name	Sexual Orientation	County
Last Name	Hometown	City
Headline	Ethnicity	Your school
Gender	Religion	Student status
Age	Smoker	Dates attended
City	Drinker	Year graduated
Country	Children	Degree type
Region	Education	Major
Zip code	Occupation	Minor
Photo link	Zodiac	Grades

Table 4.1: A Sample of Profile's Attributes

Table 4.1 contains some of the profile attributes that can be completed. Database tables and columns were built based on these attributes and their values were targeted at extraction time of profile information. Secondly, the small friendship network of the profiles created helps in understanding and testing the application faster, especially in the early stages of designing the application.

From the experimental work, we recognized two different representations of the basic information of Myspace profiles. The difference is not related to the customization feature that MySpace allows its users to apply on their profiles. It is a reflection of how data has been represented in the source document of the web page in the first place, and this affects the design of the parser to extract data from the profile webpage.

Figure 4.9 shows the basic information from two profiles which have a different appearance, while Figures 4.10 and 4.11 represent the corresponding source documents of the web pages of the two profiles. Note that the information displayed in Figure 4.9 is highlighted in Figures 4.10 and 4.11.



26 years old

Last Login:

Roga

View My: Photos Videos

25 / Female
Bradford, Northwest, UK
Last Login: 18/05/2009
Online Now!

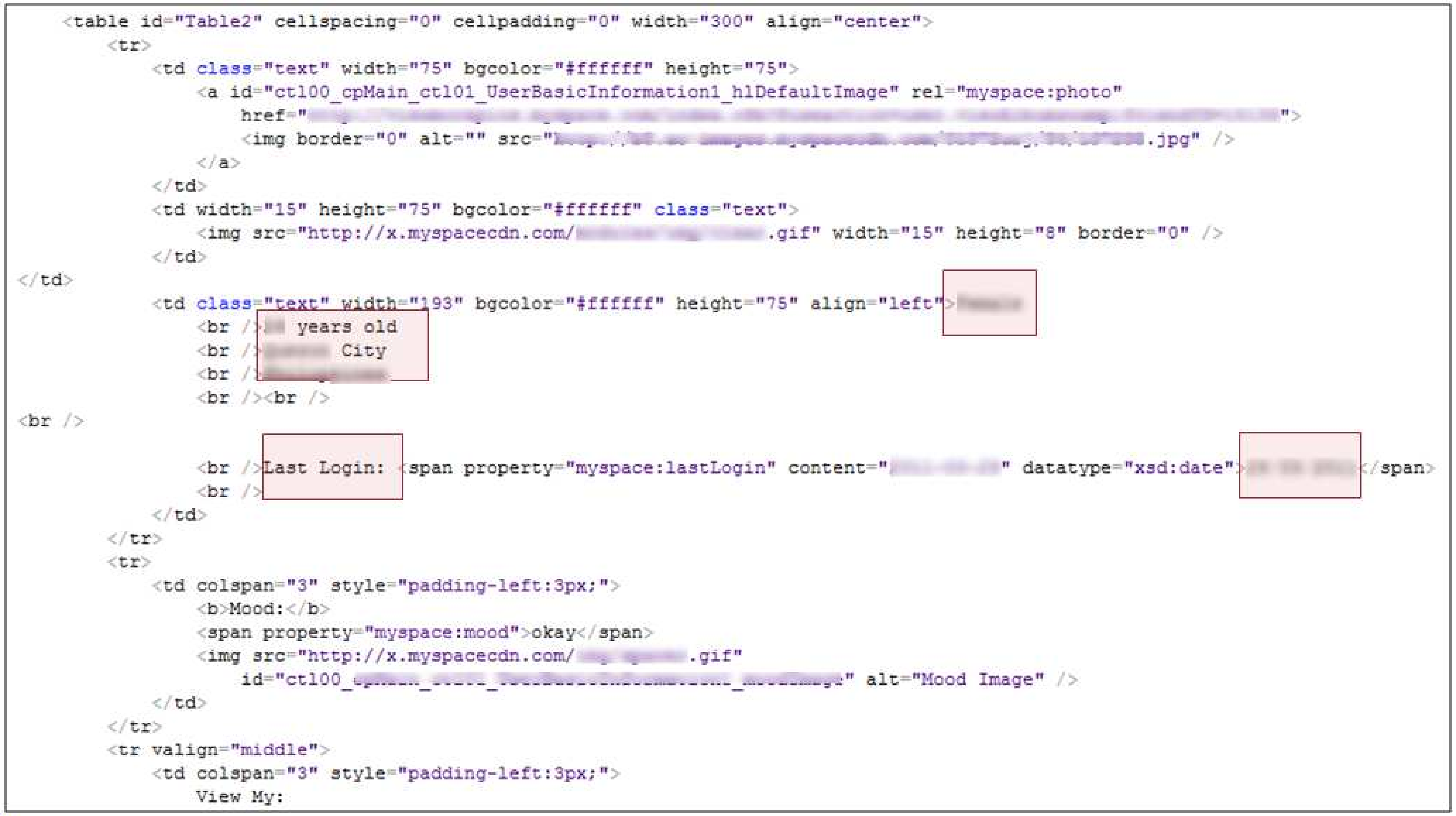
Figure 4.9: Two Different Public Profile Representations

It is obvious that in Figure 4.10, the highlighted data is unstructured, while it is semi-structured in Figure 4.11. To clarify this point, consider the attributes Gender, Age, City, Country and Region of the profiles' basic information (See Table 4.2). In order to find the values of these attributes in case of the unstructured data (Figure 4.10), the correct table of basic information must be found and then the HTML tags stripped out to find the tokens which are outside the tags but delimited by the tags as shown in the first column of Table 4.2. The tokens, which are mostly siblings in the DOM tree, will be extracted and saved in the corresponding columns of the Basic Info table in the local repository.

In contrast, in the case of semi-structured data (Figure 4.11), the tag gives a sense of what the value of the attribute would be (see Table 4.3). The parser should be able to take into account all these different structures. Thus, the parser that is presented in Figure 4.4 has been improved to retrieve the required data in the semi-structured format along with the data that is in an unstructured format.

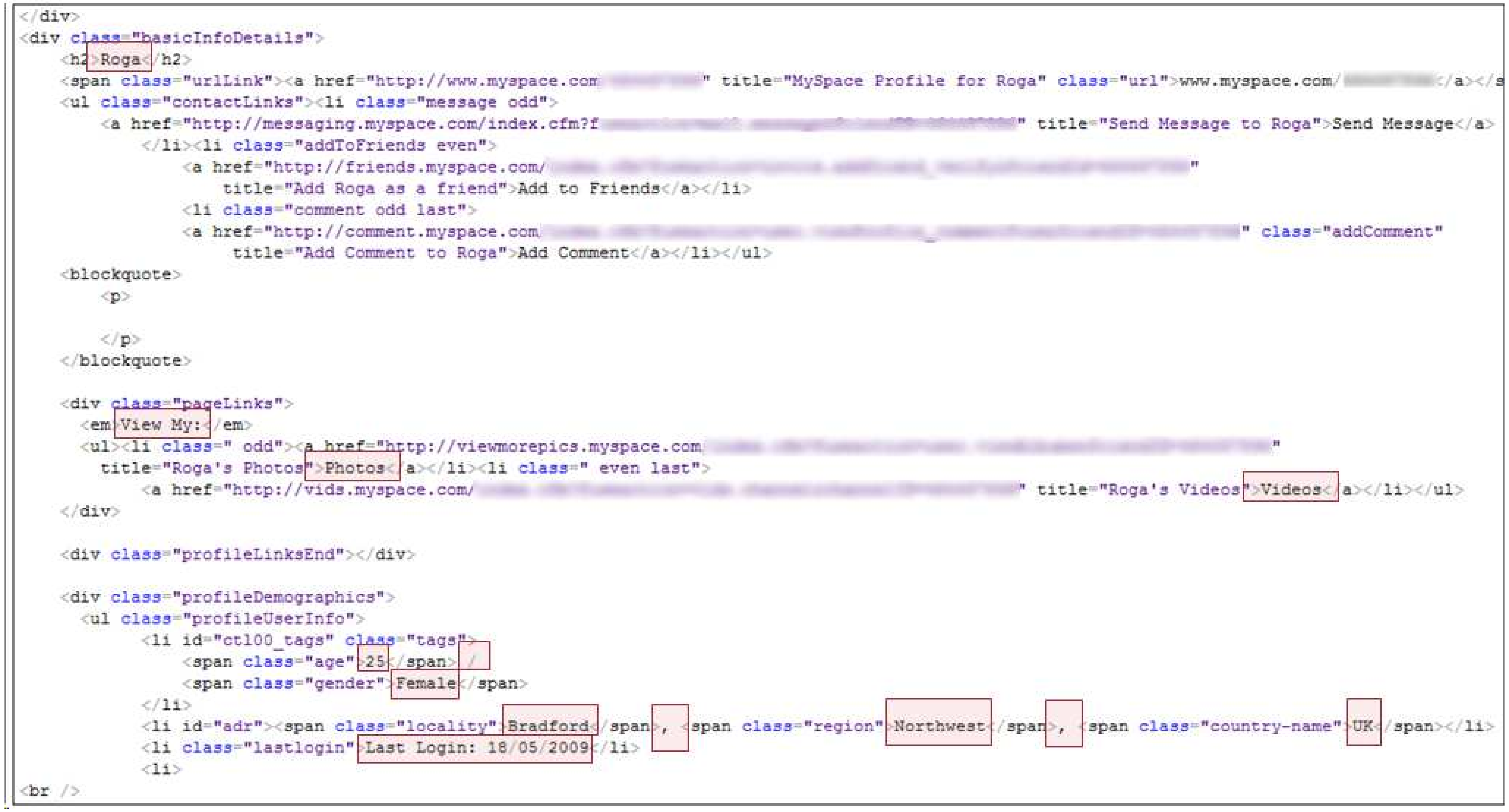
Representation of Source Data	**Attributes**	**Values**
`<tdclass="text" ... "align = "left" > Female`	Gender	Female
`<br />26 years old`	Age	26 years old
`<zr />any City`	City	any City
`<br />any Country`	Country	any Country
`<br />`	Region	null

Table 4.2: Representation of Attribute Values in an Unstructured Web Source



```
    <table id="Table2" cellspacing="0" cellpadding="0" width="300" align="center">
        <tr>
            <td class="text" width="75" bgcolor="#ffffff" height="75">
                <a id="ctl00_cpMain_ctl01_UserBasicInformation1_hlDefaultImage" rel="myspace:photo"
                    href="[redacted]">
                    <img border="0" alt="" src="[redacted].jpg" />
                </a>
            </td>
            <td width="15" height="75" bgcolor="#ffffff" class="text">
                <img src="http://x.myspacecdn.com/[redacted].gif" width="15" height="8" border="0" />
            </td>
</td>
            <td class="text" width="193" bgcolor="#ffffff" height="75" align="left">[redacted]
                <br />[redacted] years old
                <br />[redacted] City
                <br />[redacted]
                <br /><br />
<br />

                <br />Last Login: <span property="myspace:lastLogin" content="[redacted]" datatype="xsd:date">[redacted]</span>
                <br />
            </td>
        </tr>
        <tr>
            <td colspan="3" style="padding-left:3px;">
                <b>Mood:</b>
                <span property="myspace:mood">okay</span>
                <img src="http://x.myspacecdn.com/[redacted].gif"
                    id="ctl00_[redacted]" alt="Mood Image" />
            </td>
        </tr>
        <tr valign="middle">
            <td colspan="3" style="padding-left:3px;">
                View My:
```

Figure 4.10: HTML Source of a Profile Showing Some Tokens in the Unstructured Web Page



```
</div>
<div class="basicInfoDetails">
    <h2>Roga</h2>
    <span class="urlLink"><a href="http://www.myspace.com[redacted]" title="MySpace Profile for Roga" class="url">www.myspace.com/[redacted]</a></s
    <ul class="contactLinks"><li class="message odd">
        <a href="http://messaging.myspace.com/index.cfm?f[redacted]" title="Send Message to Roga">Send Message</a>
            </li><li class="addToFriends even">
                <a href="http://friends.myspace.com/[redacted]"
                      title="Add Roga as a friend">Add to Friends</a></li>
                <li class="comment odd last">
                <a href="http://comment.myspace.com[redacted]" class="addComment"
                       title="Add Comment to Roga">Add Comment</a></li></ul>
    <blockquote>
        <p>

        </p>
    </blockquote>

    <div class="pageLinks">
      <em>View My:</em>
      <ul><li class=" odd"><a href="http://viewmorepics.myspace.com[redacted]"
        title="Roga's Photos">Photos</a></li><li class=" even last">
            <a href="http://vids.myspace.com/[redacted]" title="Roga's Videos">Videos</a></li></ul>
    </div>

    <div class="profileLinksEnd"></div>

    <div class="profileDemographics">
      <ul class="profileUserInfo">
            <li id="ctl00_tags" class="tags">
                <span class="age">25</span> /
                <span class="gender">Female</span>
            </li>
            <li id="adr"><span class="locality">Bradford</span>, <span class="region">Northwest</span>, <span class="country-name">UK</span></li>
            <li class="lastlogin">Last Login: 18/05/2009</li>
            <li>
<br />
```

Figure 4.11: HTML Source Text of a Profile Showing Some Tokens in the Semi-Structured Web Page

Representation of Source Data	Attributes	Values
`<span class="gender">Female</span>`	Gender	Female
`<span class="age">25</span>`	Age	25
`<span class="locality">Bradford</span>`	City	Bradford
`<span class="country-name">UK</span>`	Country	UK
`<span class="region">Northwest</span>`	Region	Northwest

Table 4.3: Representation of Attribute Values in a Semi-Structured Web Source

4.3.2 Initializing Inputs

This stage contains two main steps:

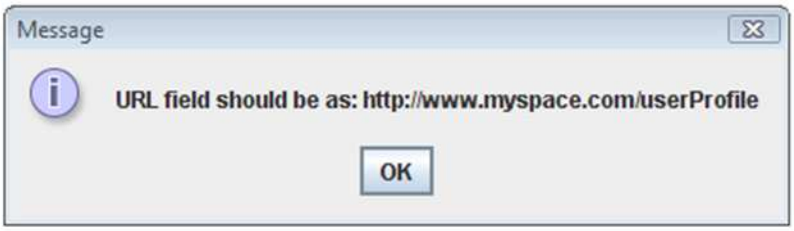
1. ***Specifying the URL Address of a Profile***

The second stage of this automated approach is to specify the URL of the seed profile to start the extraction process. All social network profiles come with a unique profile URL address. In 2009 and 2010, the URL address of Myspace profiles was clearly placed by default on the top right of the profile webpage, as shown in Figure 4.3. Then, in early 2011, it was placed in a third box on the left of the profile. At present, the URL address has been removed from the main page of the user profile by default. To find the profile URL address, the mouse is placed over the profile picture or the hyperlink of the name of the profile user, and a small pop up window appears on the bottom left of the screen.

Note that the URL address given in Figure 4.3 is one of the dummy accounts which were created for coding and testing. The account has been deleted to avoid any breach of privacy. Once the user has inserted the URL, the application will add it to a queue and validate it as it should start with `http://www.myspace.com/` followed by the unique ID. Should there be any mismatch of this format, the user will be informed and requested to correct it as shown in Figure 4.12.

2. ***Specifying the Stopping Criteria***

The users can specify whether they want to stop the extraction by either:



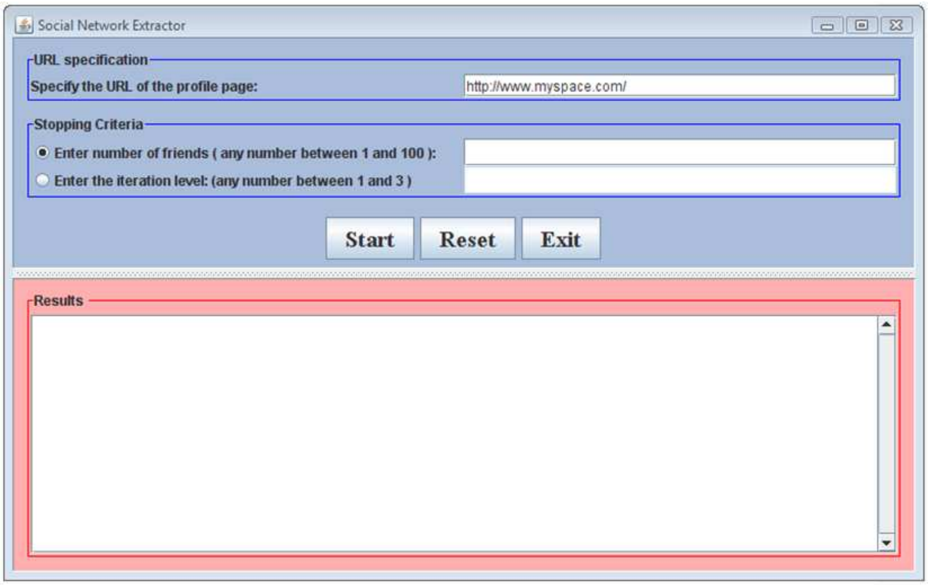
URL field should be as: http://www.myspace.com/userProfile

Figure 4.12: Error Message

1. The number of friends extracted (e.g. the first 100 friends).

2. The level of iteration (e.g. a friend of a friend). The seed URL is specified at level 0. Thus, level 1 represents the friends of the seed profile; level 2 gives the friends of the seed URL's friends, and so on. In this experiment, extraction was stopped at the third level of iteration.

A screen shot of the Graphical User Interface (GUI) of the application "Social Network Extractor" is shown in Figure 4.13. The blue section represents the information required before starting the extraction process, while the pink section represents the result text area.



URL specification
Specify the URL of the profile page: http://www.myspace.com/

Stopping Criteria
● Enter number of friends (any number between 1 and 100):
○ Enter the iteration level: (any number between 1 and 3)

Figure 4.13: Screen Shot of the Social Network Extractor Application

4.3.3 Data Collection and Processing

The third stage of the application begins by clicking the **"Start"** button as shown in Figure 4.13. The application ensures that the loop condition is verified in order to establish the following actions:

Step 1: either obtains the seed URL from the queue of friends list, if the application is running for first time, or the URL of the friends if the application is running for the next iteration.

Step 2: visits the specified URL and checks if the profile is available. This requires downloading of the HTML document as explained previously if the profile belongs to a public or private individual user.

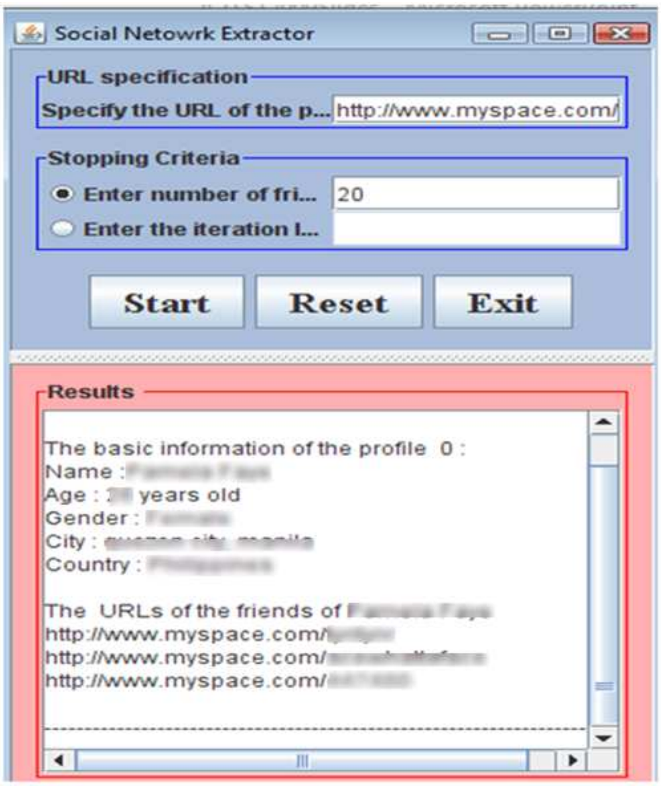
Step 3: extracts the values of the specified attributes in the parser from the data pre-processing stage. For the unstructured data, all HTML tags were stripped out from the string, the remaining text split into tokens and the tokens placed into a vector. For the semi-structured data, it is necessary to recognize what is between the tags to be used as a unique keyword. Figure 4.14 represents some of the extracted values of the basic information attributes.

Step 4: saves all extracted data to a local repository to be used for further analysis. "Postgresql[1]" has been used to create relational database tables. Mostly, each keyword specified in step 3 corresponds to a column in the database tables. The value of attributes will be extracted if it is specified by the user; otherwise, it will be null.

Step 5: extracts a list of the profile's friends. Public profiles allow a list of friends and their URL addresses to be extracted, unlike private profiles which just some of the basic information can be accessed.

As mentioned in Section 4.2.2, Myspace classifies friends list into top friends

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



Specify the URL of the p... http://www.myspace.com/

● Enter number of fri... 20

○ Enter the iteration l...

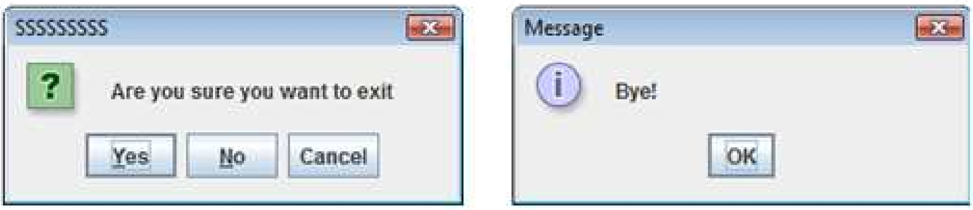
```
The basic information of the profile  0 :
Name : [redacted]
Age : [redacted] years old
Gender : [redacted]
City : [redacted]
Country : [redacted]

The  URLs of the friends of [redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
```

Figure 4.14: Screen Shot of a Sample of the Seed Profile Extracted Information

and all friends. The choice of whether to extract the profiles' top friends or all friends depends on the purpose of the extraction. A full list of friends could give a more accurate picture of the sub network and the environment of the profiles. From an extraction perspective, the algorithm to extract the profile's top friends list is similar to extracting the all friends list. A BFS algorithm is applied to each of them to select the next profile for extraction.

The difference starts when the parser must crawl through pages' hyperlinks to get the all friends list. Extra limitations have been added when dealing with all friends:

1. The profile is skipped if the number of all friends is too big, such as if the profile has 40,000 friends or hundreds of thousands of friends. E.g. Tom, the default friend when a Myspace account is created, has more than 2 million



SSSSSSSSS

Are you sure you want to exit

Message

Bye!

Figure 4.15: Confirmation before Exiting Application

friends.

2. The maximum level of iteration was set as two to minimize the sub network. The Java virtual machine heap, which is used for dynamic memory allocation, is affected and the application may even stop if the number of friends is too great.

Step 6: saves the extracted friends list in the database tables as well as saving it in a queue. Note that the URLs in the queue are unique: i.e. the URL address will be added only if it has not been stored before, to prevent duplication in visiting profiles.

As illustrated in the flowchart, the first iteration is completed when the crawler finishes extracting information of the seed URL. It will then automatically move on to obtain the next profile in the queue of friends list to extract the information of the friends and friends of friends of the seed profile, until the condition of stopping criteria are met.

The user has a choice to reset the application by pressing the **“Reset”** button in Figure 4.14 if he would like to start extracting information from another seed profile, or exit the application when the **“Exit”** button is pressed. The dialogue boxes (Figure 4.15) pop up to confirm terminating the application.

4.4 Experimental Results

Even though the purpose of the research is to present an approach to extracting data from OSN profiles automatically rather than concentrating on the extracted data itself, some interesting results have been obtained. The results will be divided into two sections according to the type of friends list; top friends and all friends. For both types, the extracted friendship network will be presented as a graph, where the profiles are the vertices and the edges represent the friendship relationship between two profiles in OSN. The open source application Nodexl[1] has been used to draw the graphs.

4.4.1 Top Friends Results

Figures 4.16 and 4.17 represent graphs of Myspace sub networks for the same seed profiles extracted at different times. The first graph of the sub network in Figure 4.16, which was extracted at the end of 2009, corresponds to the extracted information of the profile shown in Figure 4.14, where the seed profile is node 0. This node has 3 top friends in its list; nodes 1, 2 and 3. The nodes are symbolized by red spheres in Figure 4.16.

By extracting the information of node 0 and its list of top friends, first iteration is accomplished. The second iteration starts when node 1 is parsed to extract its information and list of top friends (symbolized by solid diamonds). The same process is applied to nodes 2 and 3 to complete level 1. Continuing this, 500 seed profiles have been visited; 297 public profiles were parsed to obtain 2197 top friends' profiles and 2747 friendship edges.

The second graph (Figure 4.17) was extracted in mid-2010 for the same seed profile. The sub network has grown although the seed profile has removed one of its

[1] http://nodexl.codeplex.com/

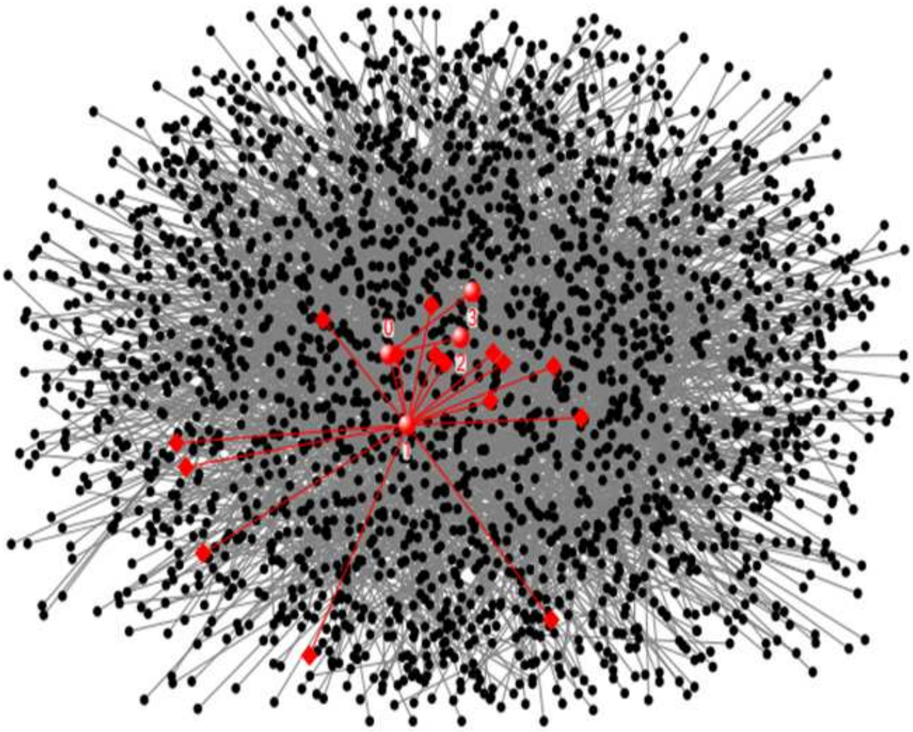


Figure 4.16: Graph 1 of a Myspace Sub Network (Top Friends)

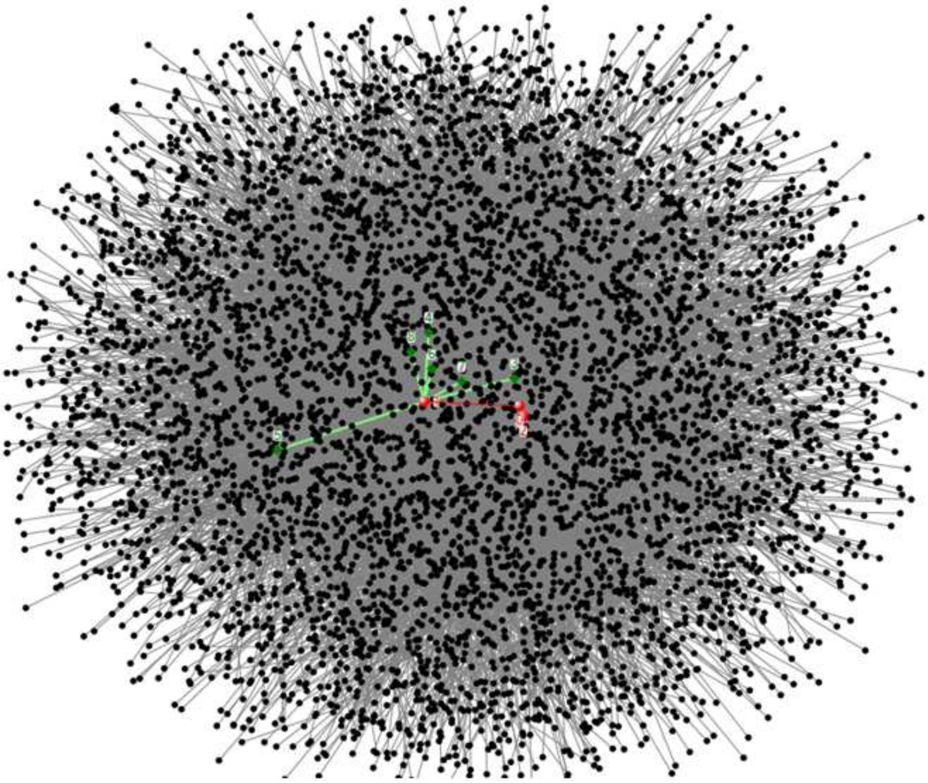


Figure 4.17: Graph 2 of a Myspace Sub Network (Top Friends)

Graph Metric	Graph1	Graph2
Vertices	2197	5395
Unique Edges	2465	6259
Edges With Duplicates	282	474
Total Edges	2747	6733
Visited Profiles	500	1300
Public Profiles	297	542

Table 4.4: Metrics of the Sub Networked Graph 1 and 2

top friends, i.e. node 0 has two friends; nodes 1 and 2. Moreover, node 2 is private, which means its friends list could not be extracted.

Table 4.4 summarizes some of the metrics of the two graphs. From the table, it can be predicted that two profiles have a strong friendship if they are in each others' list of top friends: the closest friends. Although the friendship is a symmetric relationship, top friends relationship is not: e.g. nodes A, B and C are friends. Thus each profile should be in the list of all friends of the other. However, while A and C are in the list of top friends of each other, A is in the list of B's top friends, but B is not in the list of A's top friends. Thus it can be concluded that A and C are closer to each other than B. These mutual top friends are shown in the table as duplicate edges (282 in Graph 1 and 474 in Graph 2).

While Figure 4.18 confirms the results of other researches which show that the

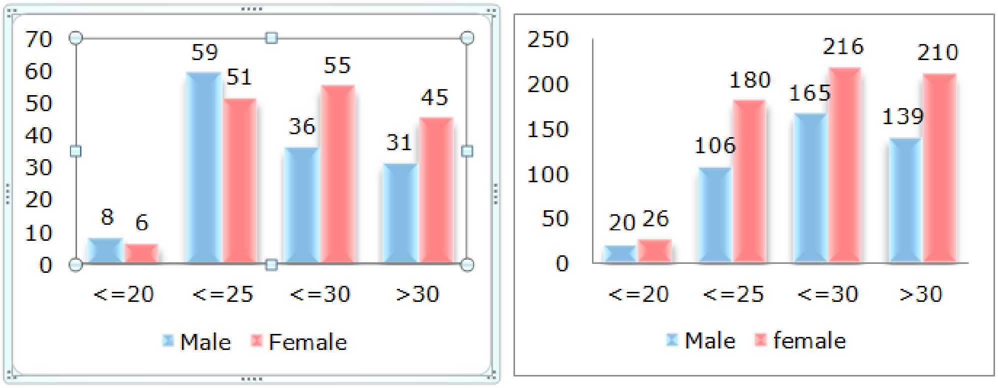


Figure 4.18: Histogram of Age for both Genders in Graph 1 and Graph 2

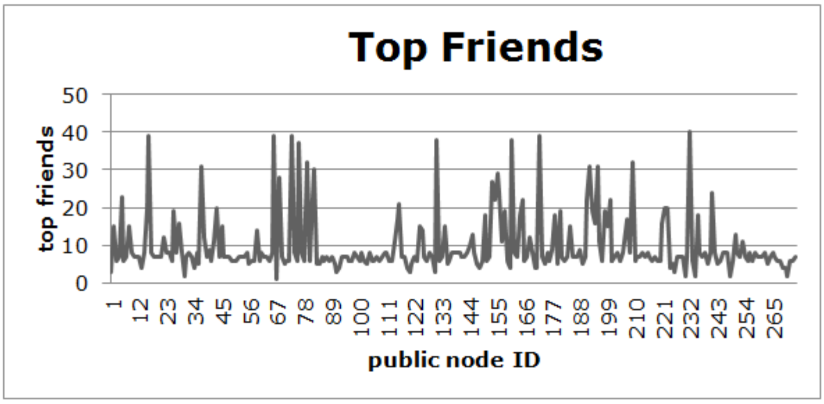


Figure 4.19: Distribution of Top Friends of Public Profiles

majority of Myspace users are female (e.g. [45]), Figure 4.19 shows the histogram of number of top friends in the public profiles. The average number of top friends is 10 while the mode is 7. The results of the extraction are used to calculate the vulnerability of friends which is detailed in [20].

4.4.2 All Friends Results

Figure 4.20 represents the graph of Myspace sub networks for the seed profile's all friends. The graph was built by parsing 250 profiles; 67 were public profiles to extract 10,196 friends' profiles and 17,223 friendship edges as shown in Table 4.5. The coloured vertices (230) are all friends of the seed profile. According to Nodexl, the graph is too large, as its vertices exceed 10,000.

From the extracted information, some basic statistics can be reached such as

Graph Metric	Graph1
Vertices	10196
Unique Edges	17203
Edges With Duplicates	0
Total Edges	17223
Visited Profiles	250
Public Profiles	67

Table 4.5: Some of Graph Metrics of All Friends

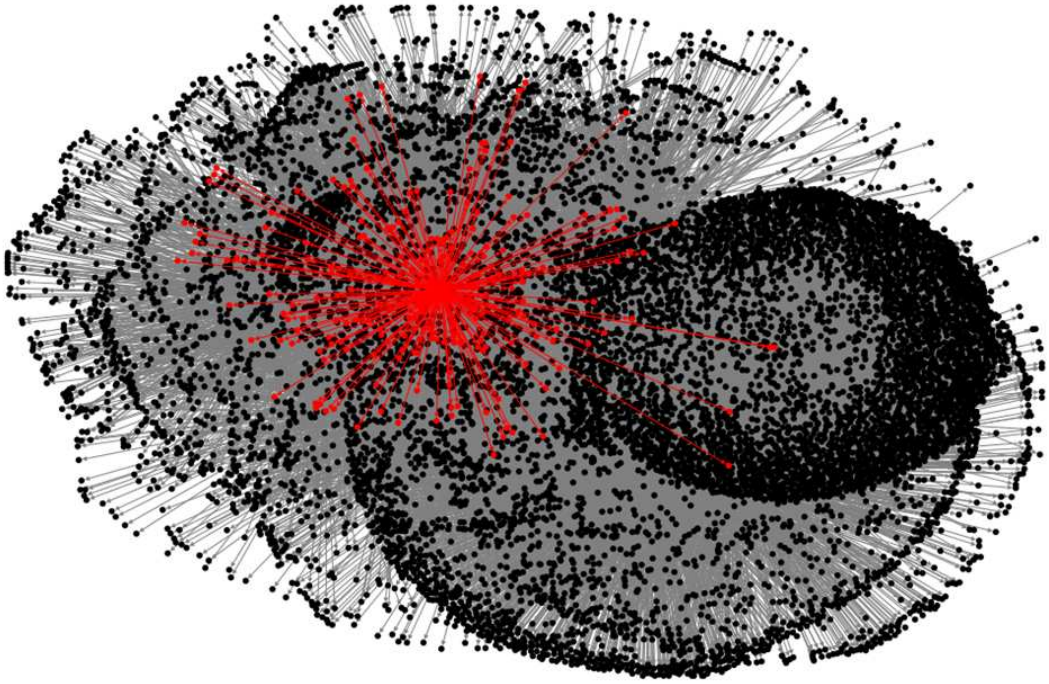


Figure 4.20: Graph of a Myspace Sub Network(AllFriends)

calculating the martial status of the users and the percentage of smokers or drinkers, as shown in Figure 4.21. Such information could be useful during analysis stages as in the results which are used to calculate the vulnerability of friends which is detailed in [9].

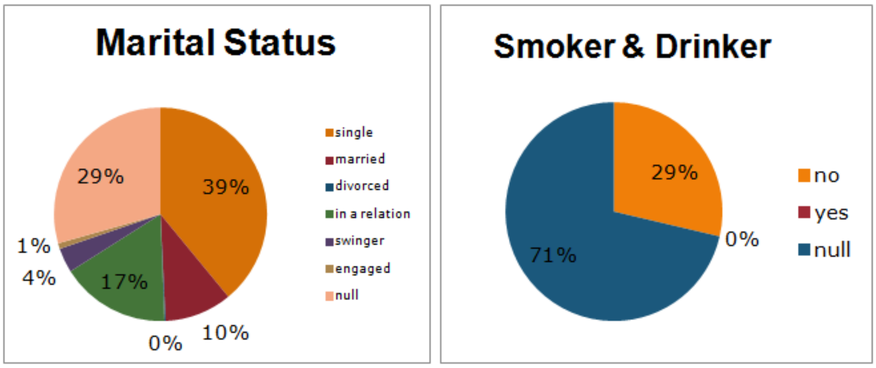


Figure 4.21: Percentages for Martial Status, Smoker and Drinker in the All Friends Sub Network

4.5 Validation

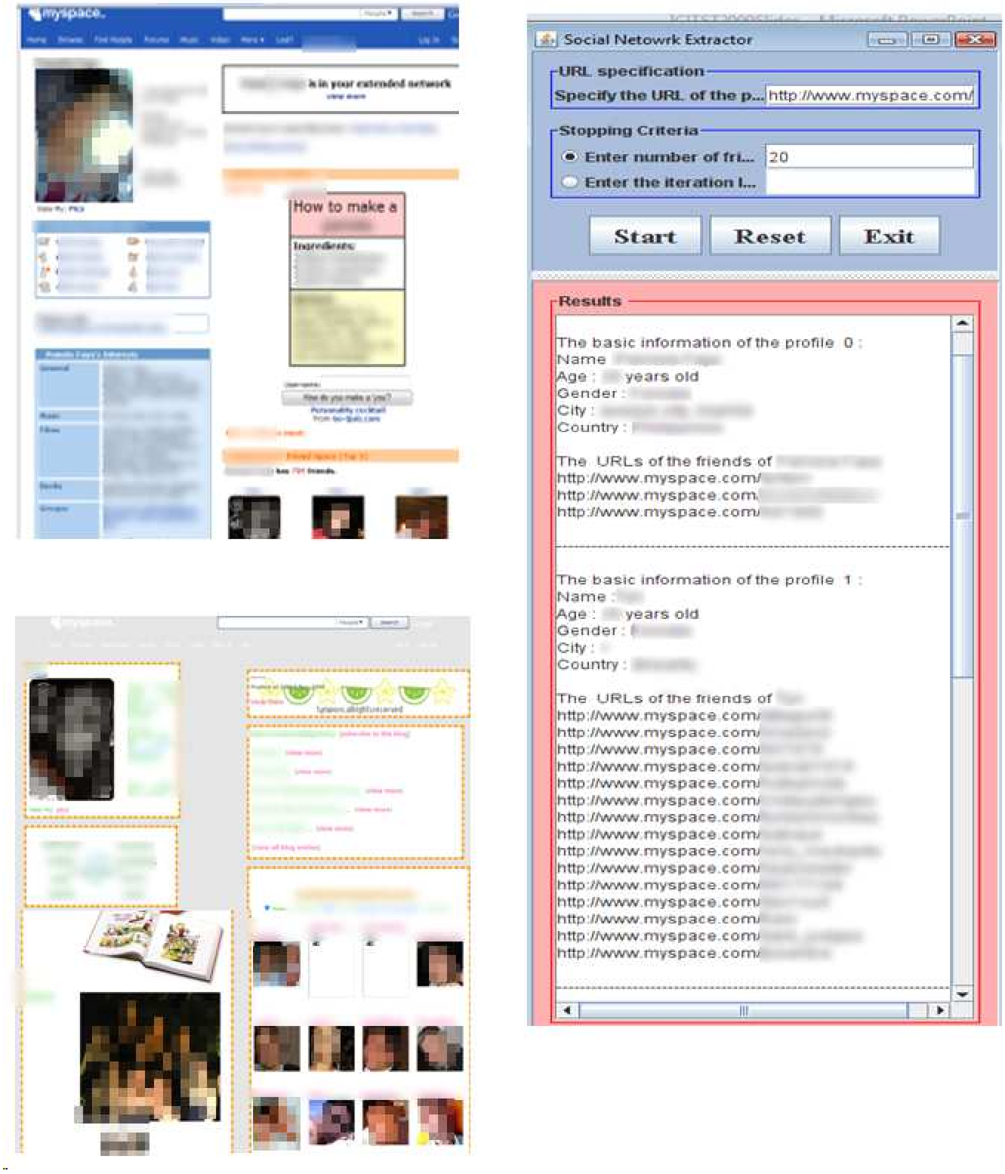
In order to confirm the accuracy of the Social Networking Extractor Application, the extracted data should undergo some form of validation. This validation process is required to ensure that the application has met the aims intended for it, which include the following:

1. Ensure that the extracted data matches what is displayed on the profile web page.
2. Ensure that the profile URL is valid, not representing one of the excluded formats and that none of the public profile has been skipped unless its friends list exceeds one thousand friends.
3. Ensure that the extracted data has been saved accurately, i.e. the attribute values are placed in the appropriate columns of the database tables.

Since it was difficult to find another application or crawler to compare results using different approaches, the results extracted by the Social Network Extractor application have been validated manually. To validate the first point, the extracted results were compared with the information displayed on the profile web page. Figures 4.22 and 4.23 present screen shots of the data extracted by the application on the right, while a screen shot of the profile web page is displayed on the left.

Figure 4.22 shows the seed profile and the first friend in its list of top friends (3 top friends), while Figure and 4.23 shows the second and the last friend.
From the figures, it should be noted that:

- The basic information displayed on the screen shots of the application results is just a part of what has been extracted by the application due to the retrieved data being too large to be displayed in its entirety.



URL specification

Specify the URL of the p... http://www.myspace.com/

Stopping Criteria

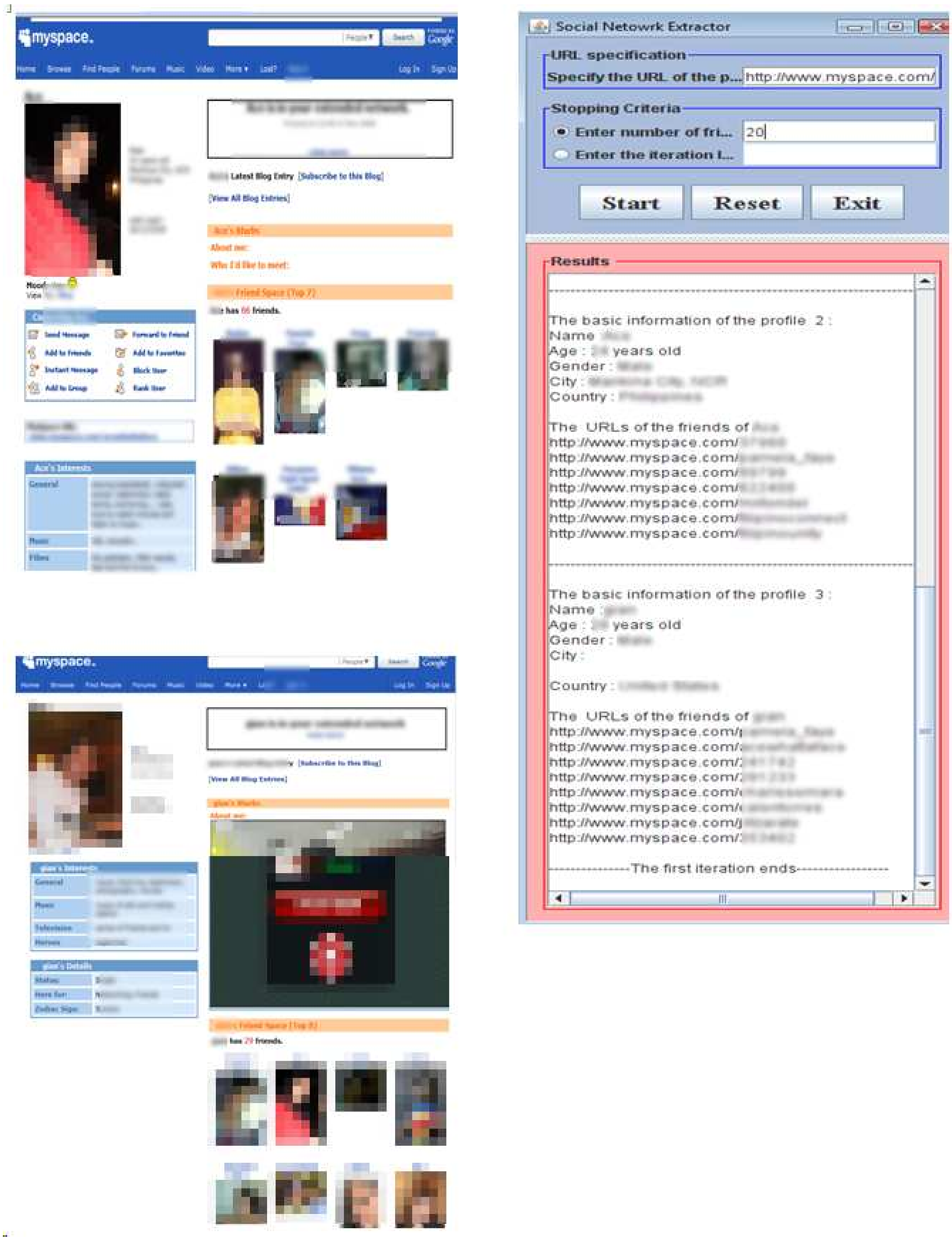
● Enter number of fri... 20

○ Enter the iteration l...

Results

```
The basic information of the profile 0 :
Name [redacted]
Age : [redacted] years old
Gender : [redacted]
City : [redacted]
Country : [redacted]

The URLs of the friends of [redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com[redacted]
http://www.myspace.com/[redacted]

The basic information of the profile 1 :
Name : [redacted]
Age : [redacted] years old
Gender : [redacted]
City : [redacted]
Country : [redacted]

The URLs of the friends of [redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
```

Figure 4.22: Manual Validation of the Seed Profile and the First Friend



Specify the URL of the p... http://www.myspace.com/

● Enter number of fri... 20

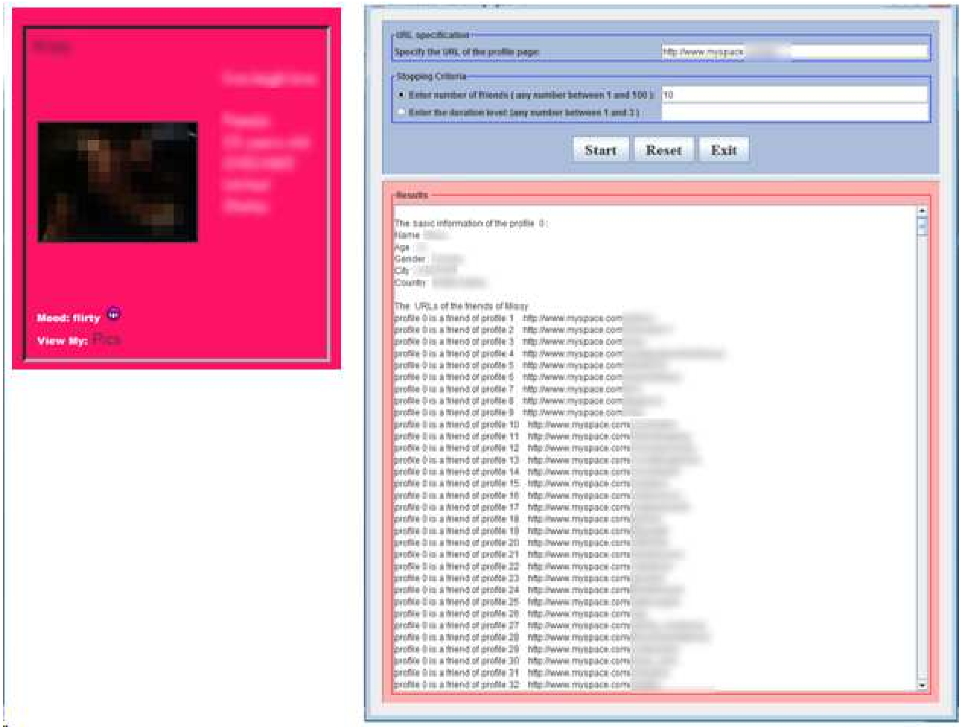
○ Enter the iteration l...

```
--------------------------------------------------------------------------------

The basic information of the profile  2 :
Name  [redacted]
Age : [redacted] years old
Gender : [redacted]
City : [redacted]
Country : [redacted]

The  URLs of the friends of [redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]

--------------------------------------------------------------------------------

The basic information of the profile  3 :
Name :[redacted]
Age : [redacted] years old
Gender : [redacted]
City :

Country : [redacted]

The  URLs of the friends of [redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]
http://www.myspace.com/[redacted]

---------------The first iteration ends-----------------
```

Figure 4.23: Manual Validation of the Second and Third Profiles

- The list of friends has been displayed completely. This is to help in ensuring that none of the public profiles is skipped unless it is one of the specified exceptions.
- There is a separator line to illustrate when the iteration level is completed. Then the crawler will start extracting the next profile in the queue, i.e. the first friend of the seed profile's first friend. The line is different than the separator lines between profiles.

Similarly, the next screen shots from Figure 4.24 to Figure 4.27 show the list of all friends for the seed profile of the graph in Figure 4.20. Note that profiles 1, 3, 4 and 5 are private profiles. Thus, just some of the basic information could be extracted, as shown in Figures 4.26 and 4.27. Screen shots of some of the private profiles' web pages are displayed in Figure 4.28.



URL specification

Specify the URL of the profile page: http://www.myspace.

Stopping Criteria

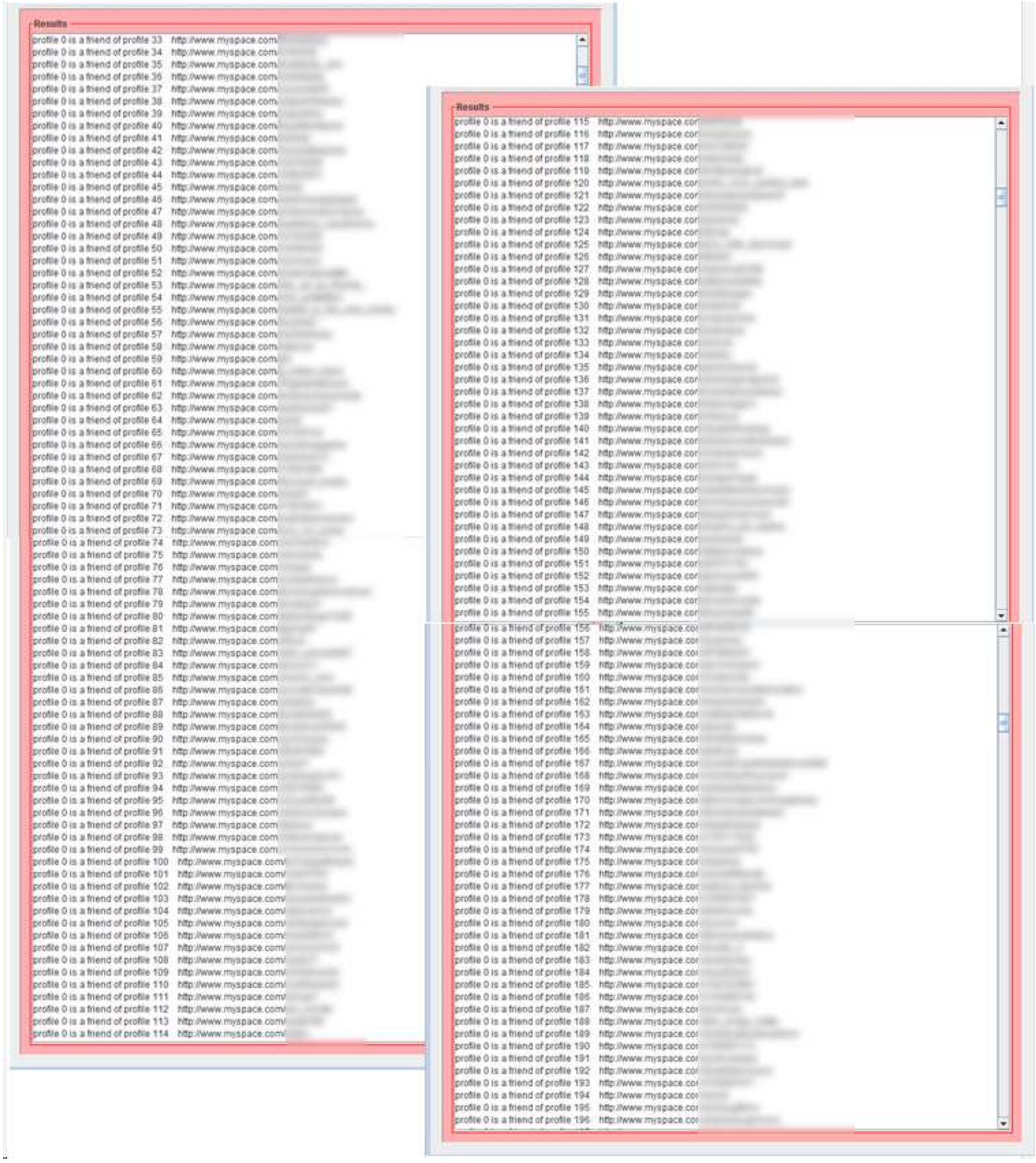
● Enter number of friends (any number between 1 and 100): 10

○ Enter the iteration level (any number between 1 and 3)

Results

```
The basic information of the profile 0 :
Name :
Age :
Gender :
City :
Country :

The URLs of the friends of Missy
profile 0 is a friend of profile 1    http://www.myspace.com/
profile 0 is a friend of profile 2    http://www.myspace.com/
profile 0 is a friend of profile 3    http://www.myspace.com/
profile 0 is a friend of profile 4    http://www.myspace.com/
profile 0 is a friend of profile 5    http://www.myspace.com/
profile 0 is a friend of profile 6    http://www.myspace.com/
profile 0 is a friend of profile 7    http://www.myspace.com/
profile 0 is a friend of profile 8    http://www.myspace.com/
profile 0 is a friend of profile 9    http://www.myspace.com/
profile 0 is a friend of profile 10    http://www.myspace.com/
profile 0 is a friend of profile 11    http://www.myspace.com/
profile 0 is a friend of profile 12    http://www.myspace.com/
profile 0 is a friend of profile 13    http://www.myspace.com/
profile 0 is a friend of profile 14    http://www.myspace.com/
profile 0 is a friend of profile 15    http://www.myspace.com/
profile 0 is a friend of profile 16    http://www.myspace.com/
profile 0 is a friend of profile 17    http://www.myspace.com/
profile 0 is a friend of profile 18    http://www.myspace.com/
profile 0 is a friend of profile 19    http://www.myspace.com/
profile 0 is a friend of profile 20    http://www.myspace.com/
profile 0 is a friend of profile 21    http://www.myspace.com/
profile 0 is a friend of profile 22    http://www.myspace.com/
profile 0 is a friend of profile 23    http://www.myspace.com/
profile 0 is a friend of profile 24    http://www.myspace.com/
profile 0 is a friend of profile 25    http://www.myspace.com/
profile 0 is a friend of profile 26    http://www.myspace.com/
profile 0 is a friend of profile 27    http://www.myspace.com/
profile 0 is a friend of profile 28    http://www.myspace.com/
profile 0 is a friend of profile 29    http://www.myspace.com/
profile 0 is a friend of profile 30    http://www.myspace.com/
profile 0 is a friend of profile 31    http://www.myspace.com/
profile 0 is a friend of profile 32    http://www.myspace.com/
```

Figure 4.24: Seed Profiles info and list of all friends



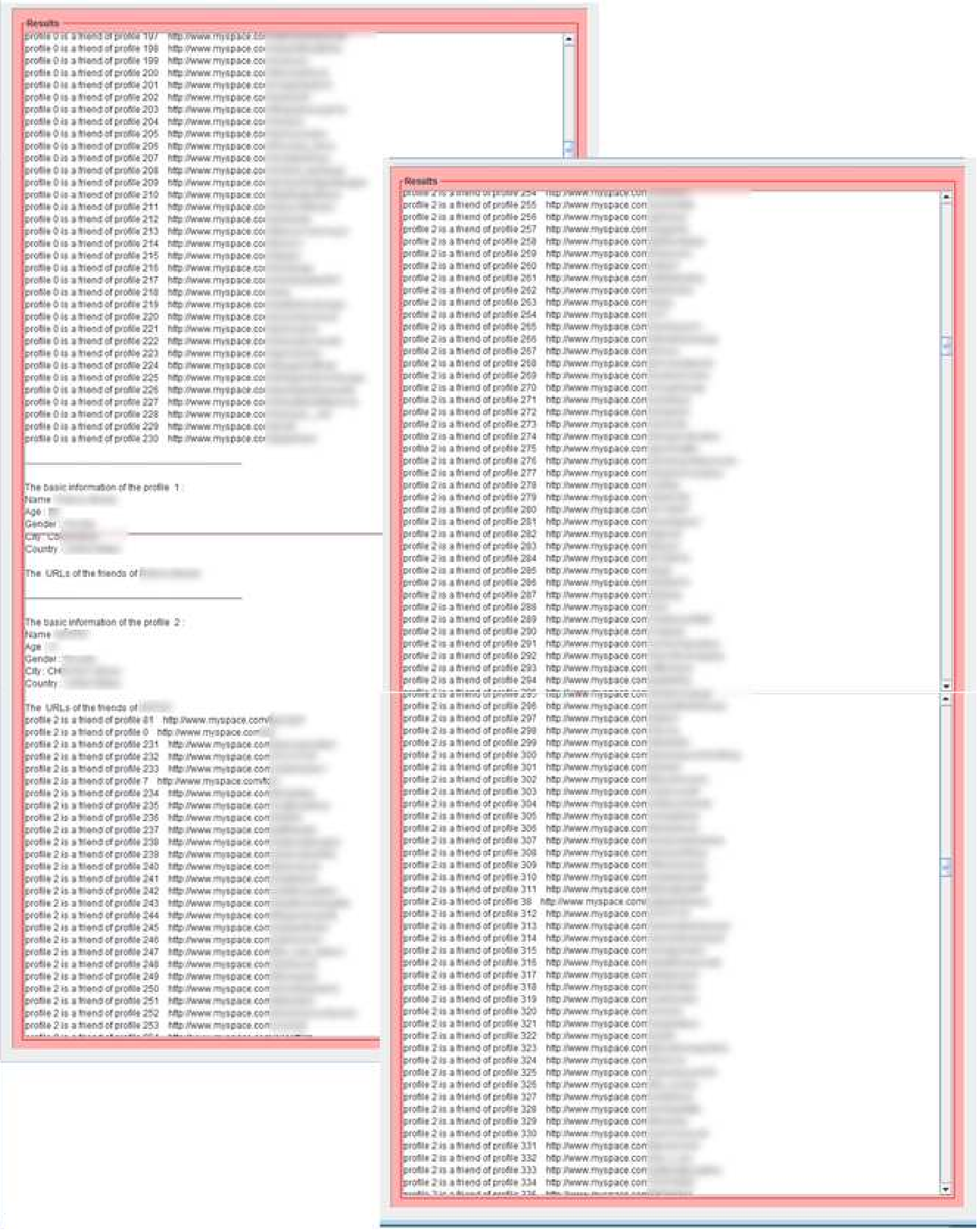
Results

```
profile 0 is a friend of profile 33   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 34   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 35   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 36   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 37   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 38   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 39   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 40   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 41   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 42   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 43   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 44   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 45   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 46   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 47   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 48   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 49   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 50   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 51   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 52   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 53   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 54   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 55   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 56   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 57   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 58   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 59   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 60   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 61   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 62   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 63   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 64   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 65   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 66   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 67   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 68   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 69   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 70   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 71   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 72   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 73   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 74   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 75   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 76   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 77   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 78   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 79   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 80   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 81   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 82   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 83   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 84   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 85   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 86   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 87   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 88   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 89   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 90   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 91   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 92   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 93   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 94   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 95   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 96   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 97   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 98   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 99   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 100   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 101   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 102   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 103   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 104   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 105   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 106   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 107   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 108   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 109   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 110   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 111   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 112   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 113   http://www.myspace.com/[redacted]
profile 0 is a friend of profile 114   http://www.myspace.com/[redacted]
```

Results

```
profile 0 is a friend of profile 115   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 116   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 117   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 118   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 119   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 120   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 121   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 122   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 123   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 124   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 125   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 126   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 127   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 128   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 129   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 130   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 131   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 132   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 133   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 134   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 135   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 136   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 137   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 138   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 139   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 140   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 141   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 142   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 143   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 144   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 145   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 146   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 147   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 148   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 149   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 150   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 151   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 152   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 153   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 154   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 155   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 156   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 157   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 158   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 159   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 160   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 161   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 162   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 163   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 164   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 165   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 166   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 167   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 168   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 169   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 170   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 171   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 172   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 173   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 174   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 175   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 176   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 177   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 178   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 179   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 180   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 181   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 182   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 183   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 184   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 185   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 186   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 187   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 188   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 189   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 190   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 191   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 192   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 193   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 194   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 195   http://www.myspace.cor[redacted]
profile 0 is a friend of profile 196   http://www.myspace.cor[redacted]
```

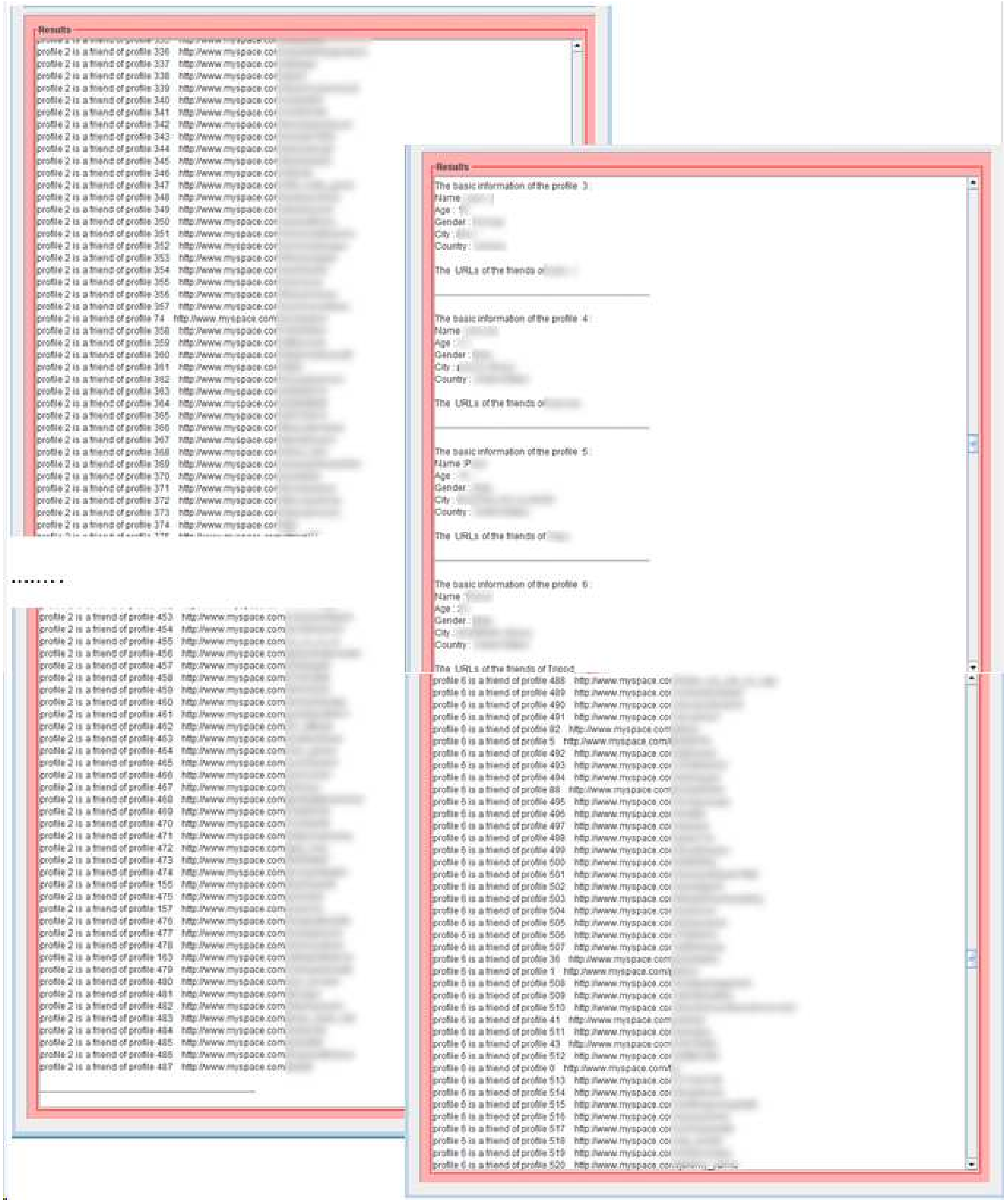
Figure 4.25: (Continued Seed Profiles list of all friends



```
Results
profile 0 is a friend of profile 197   http://www.myspace.co[redacted]
profile 0 is a friend of profile 198   http://www.myspace.co[redacted]
profile 0 is a friend of profile 199   http://www.myspace.co[redacted]
profile 0 is a friend of profile 200   http://www.myspace.co[redacted]
profile 0 is a friend of profile 201   http://www.myspace.co[redacted]
profile 0 is a friend of profile 202   http://www.myspace.co[redacted]
profile 0 is a friend of profile 203   http://www.myspace.co[redacted]
profile 0 is a friend of profile 204   http://www.myspace.co[redacted]
profile 0 is a friend of profile 205   http://www.myspace.co[redacted]
profile 0 is a friend of profile 206   http://www.myspace.co[redacted]
profile 0 is a friend of profile 207   http://www.myspace.co[redacted]
profile 0 is a friend of profile 208   http://www.myspace.co[redacted]
profile 0 is a friend of profile 209   http://www.myspace.co[redacted]
profile 0 is a friend of profile 210   http://www.myspace.co[redacted]
profile 0 is a friend of profile 211   http://www.myspace.co[redacted]
profile 0 is a friend of profile 212   http://www.myspace.co[redacted]
profile 0 is a friend of profile 213   http://www.myspace.co[redacted]
profile 0 is a friend of profile 214   http://www.myspace.co[redacted]
profile 0 is a friend of profile 215   http://www.myspace.co[redacted]
profile 0 is a friend of profile 216   http://www.myspace.co[redacted]
profile 0 is a friend of profile 217   http://www.myspace.co[redacted]
profile 0 is a friend of profile 218   http://www.myspace.co[redacted]
profile 0 is a friend of profile 219   http://www.myspace.co[redacted]
profile 0 is a friend of profile 220   http://www.myspace.co[redacted]
profile 0 is a friend of profile 221   http://www.myspace.co[redacted]
profile 0 is a friend of profile 222   http://www.myspace.co[redacted]
profile 0 is a friend of profile 223   http://www.myspace.co[redacted]
profile 0 is a friend of profile 224   http://www.myspace.co[redacted]
profile 0 is a friend of profile 225   http://www.myspace.co[redacted]
profile 0 is a friend of profile 226   http://www.myspace.co[redacted]
profile 0 is a friend of profile 227   http://www.myspace.co[redacted]
profile 0 is a friend of profile 228   http://www.myspace.co[redacted]
profile 0 is a friend of profile 229   http://www.myspace.co[redacted]
profile 0 is a friend of profile 230   http://www.myspace.co[redacted]

________________________________________

The basic information of the profile  1 :
Name [redacted]
Age : [redacted]
Gender : [redacted]
City : CO[redacted]
Country : [redacted]

The  URLs of the friends of [redacted]

________________________________________

The basic information of the profile  2 :
Name [redacted]
Age : [redacted]
Gender : [redacted]
City : CH[redacted]
Country : [redacted]

The  URLs of the friends of [redacted]
profile 2 is a friend of profile 81    http://www.myspace.com/[redacted]
profile 2 is a friend of profile 0    http://www.myspace.com/[redacted]
profile 2 is a friend of profile 231   http://www.myspace.com[redacted]
profile 2 is a friend of profile 232   http://www.myspace.com[redacted]
profile 2 is a friend of profile 233   http://www.myspace.com[redacted]
profile 2 is a friend of profile 7    http://www.myspace.com/fc[redacted]
profile 2 is a friend of profile 234   http://www.myspace.com[redacted]
profile 2 is a friend of profile 235   http://www.myspace.com[redacted]
profile 2 is a friend of profile 236   http://www.myspace.com[redacted]
profile 2 is a friend of profile 237   http://www.myspace.com[redacted]
profile 2 is a friend of profile 238   http://www.myspace.com[redacted]
profile 2 is a friend of profile 239   http://www.myspace.com[redacted]
profile 2 is a friend of profile 240   http://www.myspace.com[redacted]
profile 2 is a friend of profile 241   http://www.myspace.com[redacted]
profile 2 is a friend of profile 242   http://www.myspace.com[redacted]
profile 2 is a friend of profile 243   http://www.myspace.com[redacted]
profile 2 is a friend of profile 244   http://www.myspace.com[redacted]
profile 2 is a friend of profile 245   http://www.myspace.com[redacted]
profile 2 is a friend of profile 246   http://www.myspace.com[redacted]
profile 2 is a friend of profile 247   http://www.myspace.com[redacted]
profile 2 is a friend of profile 248   http://www.myspace.com[redacted]
profile 2 is a friend of profile 249   http://www.myspace.com[redacted]
profile 2 is a friend of profile 250   http://www.myspace.com[redacted]
profile 2 is a friend of profile 251   http://www.myspace.com[redacted]
profile 2 is a friend of profile 252   http://www.myspace.com[redacted]
profile 2 is a friend of profile 253   http://www.myspace.com[redacted]
```

```
Results
profile 2 is a friend of profile 254   http://www.myspace.com[redacted]
profile 2 is a friend of profile 255   http://www.myspace.com[redacted]
profile 2 is a friend of profile 256   http://www.myspace.com[redacted]
profile 2 is a friend of profile 257   http://www.myspace.com[redacted]
profile 2 is a friend of profile 258   http://www.myspace.com[redacted]
profile 2 is a friend of profile 259   http://www.myspace.com[redacted]
profile 2 is a friend of profile 260   http://www.myspace.com[redacted]
profile 2 is a friend of profile 261   http://www.myspace.com[redacted]
profile 2 is a friend of profile 262   http://www.myspace.com[redacted]
profile 2 is a friend of profile 263   http://www.myspace.com[redacted]
profile 2 is a friend of profile 264   http://www.myspace.com[redacted]
profile 2 is a friend of profile 265   http://www.myspace.com[redacted]
profile 2 is a friend of profile 266   http://www.myspace.com[redacted]
profile 2 is a friend of profile 267   http://www.myspace.com[redacted]
profile 2 is a friend of profile 268   http://www.myspace.com[redacted]
profile 2 is a friend of profile 269   http://www.myspace.com[redacted]
profile 2 is a friend of profile 270   http://www.myspace.com[redacted]
profile 2 is a friend of profile 271   http://www.myspace.com[redacted]
profile 2 is a friend of profile 272   http://www.myspace.com[redacted]
profile 2 is a friend of profile 273   http://www.myspace.com[redacted]
profile 2 is a friend of profile 274   http://www.myspace.com[redacted]
profile 2 is a friend of profile 275   http://www.myspace.com[redacted]
profile 2 is a friend of profile 276   http://www.myspace.com[redacted]
profile 2 is a friend of profile 277   http://www.myspace.com[redacted]
profile 2 is a friend of profile 278   http://www.myspace.com[redacted]
profile 2 is a friend of profile 279   http://www.myspace.com[redacted]
profile 2 is a friend of profile 280   http://www.myspace.com[redacted]
profile 2 is a friend of profile 281   http://www.myspace.com[redacted]
profile 2 is a friend of profile 282   http://www.myspace.com[redacted]
profile 2 is a friend of profile 283   http://www.myspace.com[redacted]
profile 2 is a friend of profile 284   http://www.myspace.com[redacted]
profile 2 is a friend of profile 285   http://www.myspace.com[redacted]
profile 2 is a friend of profile 286   http://www.myspace.com[redacted]
profile 2 is a friend of profile 287   http://www.myspace.com[redacted]
profile 2 is a friend of profile 288   http://www.myspace.com[redacted]
profile 2 is a friend of profile 289   http://www.myspace.com[redacted]
profile 2 is a friend of profile 290   http://www.myspace.com[redacted]
profile 2 is a friend of profile 291   http://www.myspace.com[redacted]
profile 2 is a friend of profile 292   http://www.myspace.com[redacted]
profile 2 is a friend of profile 293   http://www.myspace.com[redacted]
profile 2 is a friend of profile 294   http://www.myspace.com[redacted]
profile 2 is a friend of profile 295   http://www.myspace.com[redacted]
profile 2 is a friend of profile 296   http://www.myspace.com[redacted]
profile 2 is a friend of profile 297   http://www.myspace.com[redacted]
profile 2 is a friend of profile 298   http://www.myspace.com[redacted]
profile 2 is a friend of profile 299   http://www.myspace.com[redacted]
profile 2 is a friend of profile 300   http://www.myspace.com[redacted]
profile 2 is a friend of profile 301   http://www.myspace.com[redacted]
profile 2 is a friend of profile 302   http://www.myspace.com[redacted]
profile 2 is a friend of profile 303   http://www.myspace.com[redacted]
profile 2 is a friend of profile 304   http://www.myspace.com[redacted]
profile 2 is a friend of profile 305   http://www.myspace.com[redacted]
profile 2 is a friend of profile 306   http://www.myspace.com[redacted]
profile 2 is a friend of profile 307   http://www.myspace.com[redacted]
profile 2 is a friend of profile 308   http://www.myspace.com[redacted]
profile 2 is a friend of profile 309   http://www.myspace.com[redacted]
profile 2 is a friend of profile 310   http://www.myspace.com[redacted]
profile 2 is a friend of profile 311   http://www.myspace.com[redacted]
profile 2 is a friend of profile 38   http://www.myspace.com/[redacted]
profile 2 is a friend of profile 312   http://www.myspace.com[redacted]
profile 2 is a friend of profile 313   http://www.myspace.com[redacted]
profile 2 is a friend of profile 314   http://www.myspace.com[redacted]
profile 2 is a friend of profile 315   http://www.myspace.com[redacted]
profile 2 is a friend of profile 316   http://www.myspace.com[redacted]
profile 2 is a friend of profile 317   http://www.myspace.com[redacted]
profile 2 is a friend of profile 318   http://www.myspace.com[redacted]
profile 2 is a friend of profile 319   http://www.myspace.com[redacted]
profile 2 is a friend of profile 320   http://www.myspace.com[redacted]
profile 2 is a friend of profile 321   http://www.myspace.com[redacted]
profile 2 is a friend of profile 322   http://www.myspace.com[redacted]
profile 2 is a friend of profile 323   http://www.myspace.com[redacted]
profile 2 is a friend of profile 324   http://www.myspace.com[redacted]
profile 2 is a friend of profile 325   http://www.myspace.com[redacted]
profile 2 is a friend of profile 326   http://www.myspace.com[redacted]
profile 2 is a friend of profile 327   http://www.myspace.com[redacted]
profile 2 is a friend of profile 328   http://www.myspace.com[redacted]
profile 2 is a friend of profile 329   http://www.myspace.com[redacted]
profile 2 is a friend of profile 330   http://www.myspace.com[redacted]
profile 2 is a friend of profile 331   http://www.myspace.com[redacted]
profile 2 is a friend of profile 332   http://www.myspace.com[redacted]
profile 2 is a friend of profile 333   http://www.myspace.com[redacted]
profile 2 is a friend of profile 334   http://www.myspace.com[redacted]
```

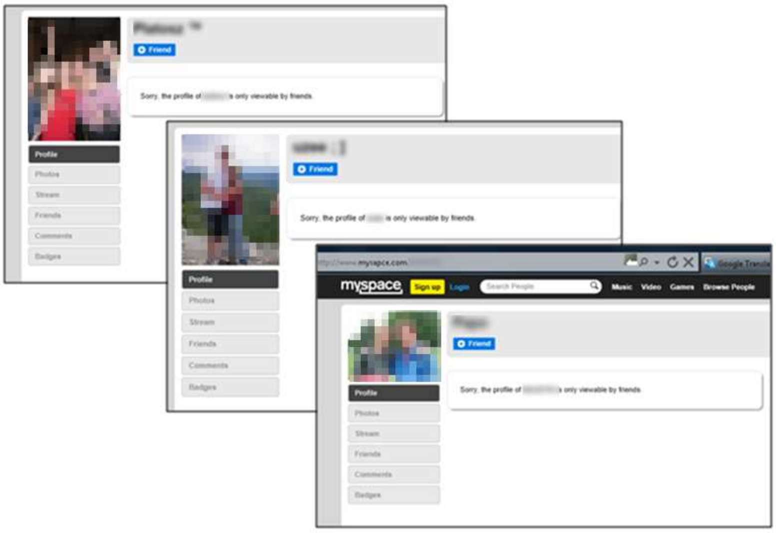
Figure 4.26: Continued Seed Profiles list of all friends



```
Results
profile 2 is a friend of profile 336   http://www.myspace.co[redacted]
profile 2 is a friend of profile 337   http://www.myspace.co[redacted]
profile 2 is a friend of profile 338   http://www.myspace.co[redacted]
profile 2 is a friend of profile 339   http://www.myspace.co[redacted]
profile 2 is a friend of profile 340   http://www.myspace.co[redacted]
profile 2 is a friend of profile 341   http://www.myspace.co[redacted]
profile 2 is a friend of profile 342   http://www.myspace.co[redacted]
profile 2 is a friend of profile 343   http://www.myspace.co[redacted]
profile 2 is a friend of profile 344   http://www.myspace.co[redacted]
profile 2 is a friend of profile 345   http://www.myspace.co[redacted]
profile 2 is a friend of profile 346   http://www.myspace.co[redacted]
profile 2 is a friend of profile 347   http://www.myspace.co[redacted]
profile 2 is a friend of profile 348   http://www.myspace.co[redacted]
profile 2 is a friend of profile 349   http://www.myspace.co[redacted]
profile 2 is a friend of profile 350   http://www.myspace.co[redacted]
profile 2 is a friend of profile 351   http://www.myspace.co[redacted]
profile 2 is a friend of profile 352   http://www.myspace.co[redacted]
profile 2 is a friend of profile 353   http://www.myspace.co[redacted]
profile 2 is a friend of profile 354   http://www.myspace.co[redacted]
profile 2 is a friend of profile 355   http://www.myspace.co[redacted]
profile 2 is a friend of profile 356   http://www.myspace.co[redacted]
profile 2 is a friend of profile 357   http://www.myspace.co[redacted]
profile 2 is a friend of profile 74   http://www.myspace.com[redacted]
profile 2 is a friend of profile 358   http://www.myspace.co[redacted]
profile 2 is a friend of profile 359   http://www.myspace.co[redacted]
profile 2 is a friend of profile 360   http://www.myspace.co[redacted]
profile 2 is a friend of profile 361   http://www.myspace.co[redacted]
profile 2 is a friend of profile 362   http://www.myspace.co[redacted]
profile 2 is a friend of profile 363   http://www.myspace.co[redacted]
profile 2 is a friend of profile 364   http://www.myspace.co[redacted]
profile 2 is a friend of profile 365   http://www.myspace.co[redacted]
profile 2 is a friend of profile 366   http://www.myspace.co[redacted]
profile 2 is a friend of profile 367   http://www.myspace.co[redacted]
profile 2 is a friend of profile 368   http://www.myspace.co[redacted]
profile 2 is a friend of profile 369   http://www.myspace.co[redacted]
profile 2 is a friend of profile 370   http://www.myspace.co[redacted]
profile 2 is a friend of profile 371   http://www.myspace.co[redacted]
profile 2 is a friend of profile 372   http://www.myspace.co[redacted]
profile 2 is a friend of profile 373   http://www.myspace.co[redacted]
profile 2 is a friend of profile 374   http://www.myspace.co[redacted]

.........

profile 2 is a friend of profile 453   http://www.myspace.com[redacted]
profile 2 is a friend of profile 454   http://www.myspace.com[redacted]
profile 2 is a friend of profile 455   http://www.myspace.com[redacted]
profile 2 is a friend of profile 456   http://www.myspace.com[redacted]
profile 2 is a friend of profile 457   http://www.myspace.com[redacted]
profile 2 is a friend of profile 458   http://www.myspace.com[redacted]
profile 2 is a friend of profile 459   http://www.myspace.com[redacted]
profile 2 is a friend of profile 460   http://www.myspace.com[redacted]
profile 2 is a friend of profile 461   http://www.myspace.com[redacted]
profile 2 is a friend of profile 462   http://www.myspace.com[redacted]
profile 2 is a friend of profile 463   http://www.myspace.com[redacted]
profile 2 is a friend of profile 464   http://www.myspace.com[redacted]
profile 2 is a friend of profile 465   http://www.myspace.com[redacted]
profile 2 is a friend of profile 466   http://www.myspace.com[redacted]
profile 2 is a friend of profile 467   http://www.myspace.com[redacted]
profile 2 is a friend of profile 468   http://www.myspace.com[redacted]
profile 2 is a friend of profile 469   http://www.myspace.com[redacted]
profile 2 is a friend of profile 470   http://www.myspace.com[redacted]
profile 2 is a friend of profile 471   http://www.myspace.com[redacted]
profile 2 is a friend of profile 472   http://www.myspace.com[redacted]
profile 2 is a friend of profile 473   http://www.myspace.com[redacted]
profile 2 is a friend of profile 474   http://www.myspace.com[redacted]
profile 2 is a friend of profile 155   http://www.myspace.com[redacted]
profile 2 is a friend of profile 475   http://www.myspace.com[redacted]
profile 2 is a friend of profile 157   http://www.myspace.com[redacted]
profile 2 is a friend of profile 476   http://www.myspace.com[redacted]
profile 2 is a friend of profile 477   http://www.myspace.com[redacted]
profile 2 is a friend of profile 478   http://www.myspace.com[redacted]
profile 2 is a friend of profile 163   http://www.myspace.com[redacted]
profile 2 is a friend of profile 479   http://www.myspace.com[redacted]
profile 2 is a friend of profile 480   http://www.myspace.com[redacted]
profile 2 is a friend of profile 481   http://www.myspace.com[redacted]
profile 2 is a friend of profile 482   http://www.myspace.com[redacted]
profile 2 is a friend of profile 483   http://www.myspace.com[redacted]
profile 2 is a friend of profile 484   http://www.myspace.com[redacted]
profile 2 is a friend of profile 485   http://www.myspace.com[redacted]
profile 2 is a friend of profile 486   http://www.myspace.com[redacted]
profile 2 is a friend of profile 487   http://www.myspace.com[redacted]
```

```
Results
The basic information of the profile  3 :
Name [redacted]
Age : [redacted]
Gender : [redacted]
City : [redacted]
Country : [redacted]

The  URLs of the friends of [redacted]
________________________________

The basic information of the profile  4 :
Name [redacted]
Age : [redacted]
Gender : [redacted]
City : [redacted]
Country : [redacted]

The  URLs of the friends of [redacted]
________________________________

The basic information of the profile  5 :
Name [redacted]
Age : [redacted]
Gender : [redacted]
City : [redacted]
Country : [redacted]

The  URLs of the friends of [redacted]
________________________________

The basic information of the profile  6 :
Name [redacted]
Age : [redacted]
Gender : [redacted]
City : [redacted]
Country : [redacted]

The  URLs of the friends of Tripod
profile 6 is a friend of profile 488   http://www.myspace.co[redacted]
profile 6 is a friend of profile 489   http://www.myspace.co[redacted]
profile 6 is a friend of profile 490   http://www.myspace.co[redacted]
profile 6 is a friend of profile 491   http://www.myspace.co[redacted]
profile 6 is a friend of profile 82   http://www.myspace.com[redacted]
profile 6 is a friend of profile 5   http://www.myspace.com/[redacted]
profile 6 is a friend of profile 492   http://www.myspace.co[redacted]
profile 6 is a friend of profile 493   http://www.myspace.co[redacted]
profile 6 is a friend of profile 494   http://www.myspace.co[redacted]
profile 6 is a friend of profile 88   http://www.myspace.com[redacted]
profile 6 is a friend of profile 495   http://www.myspace.co[redacted]
profile 6 is a friend of profile 496   http://www.myspace.co[redacted]
profile 6 is a friend of profile 497   http://www.myspace.co[redacted]
profile 6 is a friend of profile 498   http://www.myspace.co[redacted]
profile 6 is a friend of profile 499   http://www.myspace.co[redacted]
profile 6 is a friend of profile 500   http://www.myspace.co[redacted]
profile 6 is a friend of profile 501   http://www.myspace.co[redacted]
profile 6 is a friend of profile 502   http://www.myspace.co[redacted]
profile 6 is a friend of profile 503   http://www.myspace.co[redacted]
profile 6 is a friend of profile 504   http://www.myspace.co[redacted]
profile 6 is a friend of profile 505   http://www.myspace.co[redacted]
profile 6 is a friend of profile 506   http://www.myspace.co[redacted]
profile 6 is a friend of profile 507   http://www.myspace.co[redacted]
profile 6 is a friend of profile 36   http://www.myspace.com[redacted]
profile 6 is a friend of profile 1   http://www.myspace.com/[redacted]
profile 6 is a friend of profile 508   http://www.myspace.co[redacted]
profile 6 is a friend of profile 509   http://www.myspace.co[redacted]
profile 6 is a friend of profile 510   http://www.myspace.co[redacted]
profile 6 is a friend of profile 41   http://www.myspace.com[redacted]
profile 6 is a friend of profile 511   http://www.myspace.co[redacted]
profile 6 is a friend of profile 43   http://www.myspace.com[redacted]
profile 6 is a friend of profile 512   http://www.myspace.co[redacted]
profile 6 is a friend of profile 0   http://www.myspace.com/[redacted]
profile 6 is a friend of profile 513   http://www.myspace.co[redacted]
profile 6 is a friend of profile 514   http://www.myspace.co[redacted]
profile 6 is a friend of profile 515   http://www.myspace.co[redacted]
profile 6 is a friend of profile 516   http://www.myspace.co[redacted]
profile 6 is a friend of profile 517   http://www.myspace.co[redacted]
profile 6 is a friend of profile 518   http://www.myspace.co[redacted]
profile 6 is a friend of profile 519   http://www.myspace.co[redacted]
profile 6 is a friend of profile 520   http://www.myspace.co[redacted]
```

Figure 4.27: Continued Seed Profiles list of all friends



Sorry, the profile of [redacted] is only viewable by friends.

Sorry, the profile of [redacted] is only viewable by friends.

Sorry, the profile of [redacted] is only viewable by friends.

Figure 4.28: Sample of Private Myspace Profiles

4.6 Limitations of the Current Approach

The limitations of the current approach include:

1. When the URL of a profile is given, the parser must check the type of profile. Those profiles which represent bands, magazines and gangs are excluded because they do not represent friendship between two individuals. Private profiles are also excluded as they prevent the parser from retrieving most of their information. This may affect the structure of the sub network.

2. The structure of the profiles is unstable and changes over time: this requires modification of the parser to keep pace with the updates. The update process is time- and effort- consuming.

3. The application is built as a centralized approach. The application is likely to stop if one of the profile structures does not match the cases in the parser.

4.7 Chapter Summary

This chapter presents an automated approach for extracting data from OSN sites. The chapter concentrates on the extraction of semi- structured and unstructured data from Myspace profiles. A detailed description of the "Social Network Extractor" application has been shown. The application is implemented to extract Myspace profiles information as well as list of top friends and all friends. Some of interesting results of the extracted data are shown accompanied with graphs.

The research has shown how far social network extraction has come since the days when extracting attributes involved much of interaction with the profile owner, e.g. questionnaires and interviews. Automatic extraction of attributes is the way forward and it can be applied to dynamic web pages. The main challenge when carrying out the experiment to implement the approach was that with social networks such as MySpace, profile structures change fast over time, besides having more than one template through which the user can customize a profile.

The contribution to this chapter has been published in [20, 9, 19]. and presented in:

- The Institution of Engineering and Technology (IET) prestige invited talk: *Algorithms for Social Engineering in Online Social Network* (with S. Alim, D. Neagu and M. Ridley), (2010) University of Bradford, UK. `http://www.theiet.org/local/uk/yorks/west/social-eng.cfm`

- Open Day: *Automated Data Retrieval from Online Social Network Profiles* (with S. Alim), (2010) University of Bradford, UK.

- Presentation at FAIRS2009 for: the 3rd annual forum for AI research students: *Algorithm for Data Retrieval from Online Social Network Profiles* (2009) Cambridge University, UK.

- Presentation and Demonstration to students from Bradford Grammar School: *Data Extraction from Online Social Network Profiles* (2009) University of Bradford, UK.
- AI Research Seminar: *Data Retrieval from Online Social Networking Profiles for Social Engineering Applications* (2009) University of Bradford, UK.

Chapter 5

Formal Specification of MAS for Historical Data Extraction from OSN

Preamble

The approach developed in Chapter 4 for extracting data from OSN is a centralized system which controls and retrieves information from each user's profile once. To overcome the limitations of that approach, MAS was applied because of its ability to monitor the users' profiles continuously in a parallel paradigm. However, MAS is considered to be a complex system to develop. This chapter investigates through formal specification (Object-Z) the feasibility of using MAS technology in extracting historical data from OSN sites, to ensure that the proposed Online Social Network Retrieval System (OSNRS) is robust, reliable and fit for purpose before moving to the implementation phase.

5.1 Introduction

Coping with real time changes in a huge amount of personal information requires an adaptation of previous methods for data extraction. In the previous chapter, an approach for automated data extraction from OSN was presented. However, the approach has several limitations which need to be acknowledged. The main cause of those limits lies in the fact that the system was designed in the manner of a centralized system which uses a single agent to collect data from OSN. As a result, the system was unable to continue if one of the profiles could not be parsed correctly. Moreover, the previous algorithm did not take into account the monitoring of users' profiles continuously over time.

To overcome these limitations, an improvement to the previous approach is required to move from a single agent to MAS in order to extract and monitor updates in many profiles concurrently. As described in Chapter 2, a MAS has characteristics such as mobility, autonomy, sociability and perceptivity which enable it to work in a parallel (multithread) approach.

This chapter highlights the feasibility of using MAS technology in extracting data from OSN sites. Due to MAS is considered to be a complex system to develop, a formal specification language (Object-Z) is used to specify the enhanced web based system OSNRS in order to ensure that the system is robust, reliable and fits the purpose, before implementing the application.

The rest of the chapter is as follows: Section 5.2 presents an overview of OSNRS. The flow of information between the OSNRS components is described in Section 5.3, while Section 5.4 presents the formal specification of OSNRS. Two main classes of OSNRS: MasterAgent and GrabberAgent, are detailed in Sections 5.5 and 5.6 respectively. Finally, the summary is presented in Section 5.7.

5.2 Overview of the Enhanced System (OSNRS)

In order to have a better understanding of the formal specification of the enhanced system, an informal overview of the system components will be presented in addition to a description of how these components interact with each other. This is because the level of the informal description detail will influence the level of formal specification abstraction [55](p. 4).The enhanced system (OSNRS) can be defined as described in Definition 1.

Definition 1: OSNRS

A MAS that consists of a finite set of grabber agents controlled by a special agent called MasterAgent in order to achieve the goal of retrieving and monitoring OSN profile information starting from a given OSN profile (seed profile).

Through this thesis, the MasterAgent (which is denoted by *mAg*) organizes the extraction process, controls agents and saves the retrieved information in the local repository. In contrast, the grabber agent (denoted by *gAg*) is used to refer to the agent that is responsible for extracting data from OSN profiles and detecting updates in these profiles. Figure 5.1 shows the structural model of the enhanced system.

The general features of MAS were described in Section 2.5.5. How these features implemented in OSNRS is described briefly below and will be detailed in next chapter at Section 6.2.3.

The autonomy feature of each agent is implemented through working as a standalone process in extracting data from a specific profile and monitoring its updates. Also, the autonomy feature of *gAg* allows it to decide on a suitable period of time after which to reactivate itself depending on how active the profile is.

The sociability feature is applied when agents communicate with each other (in addition to users) and exchange their knowledge in order to achieve their main shared goal, which is to monitor the OSN profiles over time. The perceptivity feature of *gAg* is shown when it can detect the real time updates of the profile that it is assigned to whenever it is activated. More details of the OSNRS agents' features will be

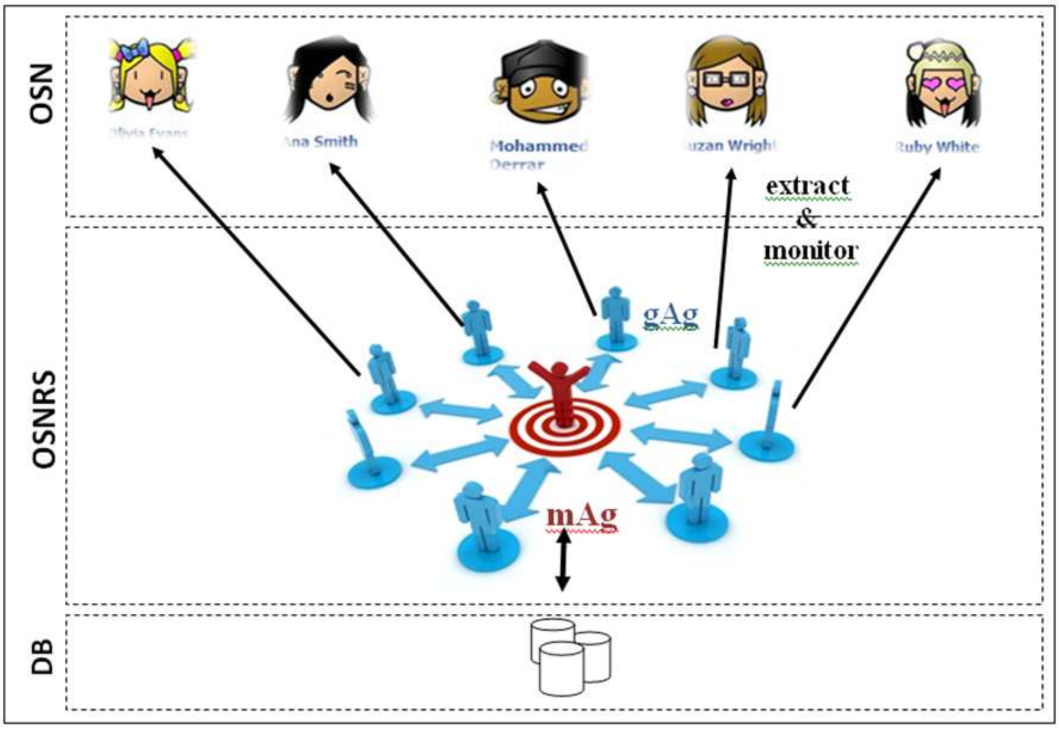


Figure 5.1: Structural Model of OSNRS

illustrated in the next chapter.

5.3 The Flow of Data between OSNRS Components

As illustrated in Figure 5.2, the OSNRS starts when the user specifies an identity of OSN to extract its information. The *mAg* receives this identity (seed profile) and adds it to the front of friends' list queue to be assigned to a *gAg* . The *gAg* browses the web page of the profile and extracts basic information such as name, gender, age, country, etc. as well as the profile's list of friends. This information will be stored locally in a file to be compared with the information retrieved the next time the *gAg* is activated.

Another copy of the file will be sent to the *mAg* and used to create a history of that profile. The *mAg* adds the set of files to a local repository to be mined later

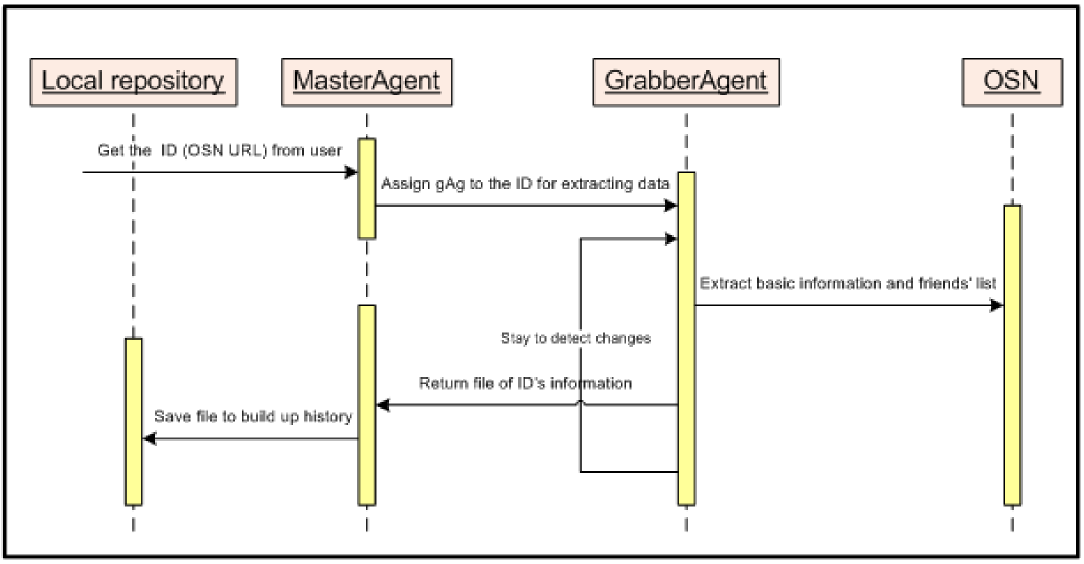

Local repository
MasterAgent
GrabberAgent
OSN
Get the ID (OSN URL) from user
Assign gAg to the ID for extracting data
Extract basic information and friends' list
Stay to detect changes
Return file of ID's information
Save file to build up history

Figure 5.2: Sequence Diagram of OSNRS

for future analysis.

Consequently, the *mAg* will add the list of friends' identities, which have not added before at the rear of the queue. Each identity will be assigned to a *gAg* , provided this has not already been done. The extraction cycle will repeat until the *mAg* matches one of the stopping criteria described earlier in Section 4.3.2. The perceptivity feature of *gAg* allows it to detect updates in the profile that it is assigned to when it is activated periodically. The updates are detected by comparing the current extracted data with the saved file of the previous extraction iteration. Once a change is captured, the *gAg* will inform the *mAg* and send a copy of the changes.

5.4 Formal Specification of OSNRS

The above informal description of the OSNRS will be expressed as a formal specification using Object-Z specification language as Hilaire et.al in [76] and Hayes in [70] selected to formalize MAS. Object-Z supports all MAS features covered by the proposed system (OSNRS) specifically concurrency, communications and state.

Also multiple communication could be applied through range of composition operators [40]. This fits well with our proposed system (OSNRS) that will use Java to implement MAS.

Initially, the basic types of OSNRS and the relations between these types will be identified. Then, the Object-Z Classes of OSNRS will be detailed independently.

5.4.1 OSNRS Basic Types

From the description of the OSNRS, the basic types of the enhanced system as shown in Figure are defined as:

$[Agent, Record, Identity]$

where:

$[Agent]$ the set of all created agents.

$[Record]$ the set of files which contains the retrieved information from the users' profiles excluding the list of friends. Time could be included in this type to specify the time of retrieving, updating and saving information.

$[Identity]$ the set of the URL addresses of OSN profiles.

As seen in Figure 5.3, the main relations between the basic types of the system are:

- *visiting*: is a partial injective function from *Agent* to *Identity* to record which identity a *gAg* is visiting. *visiting* is a partial function, to exclude the initial state and the cases where the identities are retrieved but not assigned to any *gAg* . It is injective (one-to-one) function because each identity is visited by at most one *gAg* .
- *file*: is also a partial injective function from *Identity* to *Record* to represent one record of the identity's extracted data at a time.

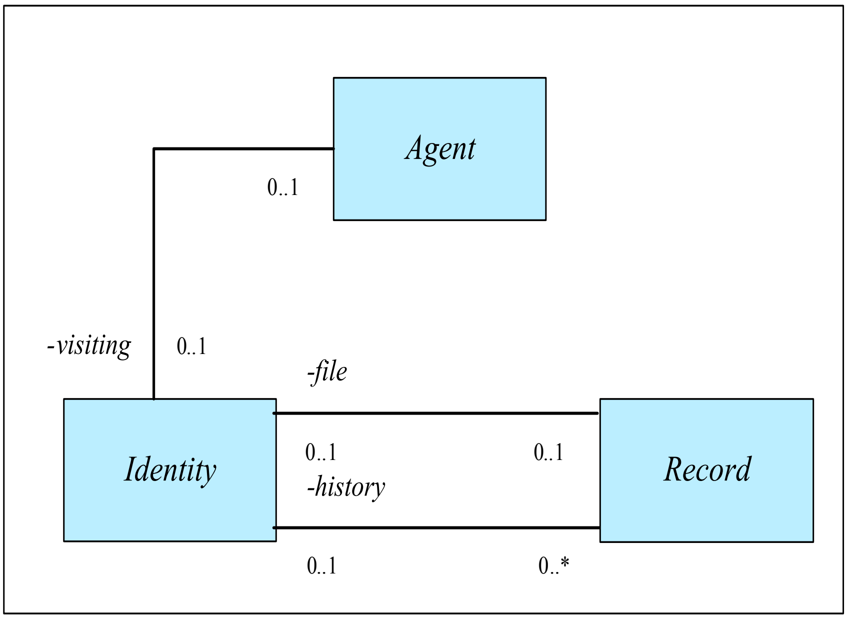


Figure 5.3: Relationships between the Basic Types of OSNRS

- *history*: records a history of each identity that has been observed over time.

5.4.2 The Object-Z Classes of OSNRS

Although many classes are compounded to construct OSNRS, as are illustrated in Figure 5.2, only the most two important classes will be focused on in the following sections:

- The MasterAgent Class.
- The GrabberAgent Class.

The syntax of these Object-Z classes may contain a constant definition, a state schema, an initial state and finally a set of operation schemas, as described in the next sections.

Master Agent
$-iteration$ $-counter$ $-known$ $-waitingresponse$ $-visiting$ $-current$ $-next$
$+INIT()$ $+getFirstidentity$ $+start()$ $+addAssigning()$ $+assignGrabber()$ $+receiveAllProfiles()$ $+nextlevel()$ $+completeReceiving()$

Figure 5.4: Class Diagram of MasterAgent

5.5 MasterAgent Class

The MasterAgent class is the main class in OSNRS which controls the extraction process as well as saving a history of extracted profile information in a local repository, as mentioned earlier. A class diagram which displays the attributes and operations of the class is shown in Figure 5.4.

Each component of the class is described in detail in the sub sections below:

5.5.1 Constants Definitions

The class MasterAgent has three constants: *iterationLevel* and *counter*. Both are natural numbers specified by the user to declare the stopping criteria of the system. The third constant *null* of type *Record* is required later in initializing the state variable of the GrabberAgent class.

$$iterationLevel : \mathbb{N}$$

$$counter : \mathbb{N}$$

$$null : \text{Record}$$

5.5.2 State Schema of MasterAgent

$$
\begin{array}{|l}
\textit{StateSchema} \\
\hline
known : \mathrm{seq}\, Identity \\
waitingResopnse : \mathbb{P}\, Identity \\
visiting : Agent \rightarrowtail\!\!\!\!\!\!\leftrightarrow Identity \\
current, next : (\mathrm{seq}\, Identity \times \mathbb{Z}) \\
\hline
waitingResponse \subseteq \mathrm{ran}\, known \\
\mathrm{ran}\, first~~next \subseteq \mathrm{ran}\, known \\
\mathrm{ran}\, first~~current \subseteq \mathrm{ran}\, known \\
\mathrm{disjoint}\langle \mathrm{ran}\, first~~current, \mathrm{ran}\, first~~next \rangle \\
\hline
\end{array}
$$

The state schema in the class MasterAgent contains five state variables as illustrated below:

- *known*: is an injective sequence of all identities (URLs) which are known to the *mAg* : either the identity has been retrieved as a friend in the friends list or entered by the user. Each identity in *known* appears once.

- *waitingResponse*: is a set of identities that *gAg* (s) have been assigned to, but whose details have not yet been retrieved.

- *visiting*: is a partial injective function from *Agent* to *Identity* as described earlier in Section 5.4.1.

- *current* and *next*: are tuples. *current* is used to determine the sequence of identities that are being retrieved in this iteration level, while *next* represents the identities that will be retrieved in the next iteration level. See Definition 2 for the recursive definition of level. In other words, when the extraction

Definition 2: Recursive definition of level

level? : * *seed*? is at level 0;

* if i is an identity at level n then,

friends of i , who have not previously been recorded are at level n +1.

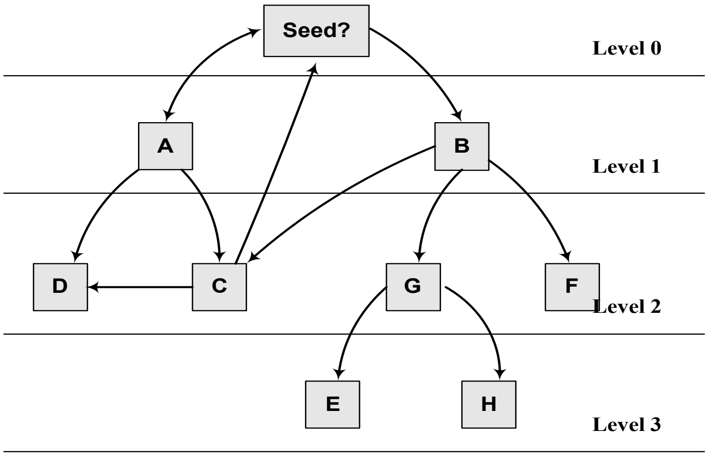
process moves from an iteration level to another level, the contents of *next* will become *current*; then *next* will contain the list of friends of identities in *current*.

The state invariants point out two roles of *known*; first of all, it has to record all retrieved identities in addition to the initial identity provided by the user, denoted (*seed*?). The second role is to filter the retrieved identities before adding them to the existing list. Thus, all identities in *next*, *current* and *waitingResponse* are taken from *known*.

At any level, the identities that are in *next* will become *current* for the next levels, and the final invariant will ensure that the identity will not be extracted twice by different *gAg*s, as will be illustrated in Figure 5.5.

Informal Description Example

To simplify understanding of the OSNRS state schema, it is useful to consider the graph and the table in Figure 5.5. The node *seed*? will be added at level 0 to *known* as shown in the table of the Figure 5.5. This node represents the root and it will be in *next* to determine that the tuple will become the *current* tuple for extraction in the level 1.



Iteration Level	*current'*	*known'*	*next'*
Initial	$(\langle\rangle, -1)$	$\langle\rangle$	$(\langle\rangle, -1)$
0	$(\langle\rangle, -1)$	$\langle seed?\rangle$	$(\langle seed?\rangle, 0)$
1	$(\langle seed?\rangle, 0)$	$\langle seed?\rangle ^\frown \langle A,B\rangle$	$(\langle A,B\rangle, 1)$
2	$(\langle A,B\rangle, 1)$	$\langle seed?,A,B\rangle ^\frown \langle D, C, G,F\rangle$	$(\langle D, C, G ,F\rangle,2)$
3	$(\langle D, C, G ,F\rangle,2)$	$\langle seed?,A,B,D, C, G,F\rangle ^\frown \langle E,H\rangle$	$(\langle E,H\rangle, 3)$

Figure 5.5: A Sample Model of Online Social Network

The node *seed*? has two top friends in its list which are A and B. These two nodes will be added to the *known* and compose the tuple to be extracted in level 2. Node A has three friends in its top friends list, which are C, D and *seed*?. Therefore, the link between *seed*? and A is bidirectional, i.e. they are in each others' top friends list. However, *known* will filter the retrieved identities to not allow any redundancy, i.e. *seed*? will not be added. The same principle is applied to the subsequent levels.

5.5.3 The Initial Schema

When an object of the class MasterAgent is initialized, no identity has been added to the sequence *known* yet. Consequently, no identities are waiting for response in *waitingResponse*.

The *visiting* variable is an empty set because the *gAg* has not been allocated to any identity. Therefore, *current* and *next* are tuples that, on level -1, contain empty sequences of identities which are being or will be retrieved respectively.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\mathit{INIT}} \\
\hline
\mathit{known} = \langle\,\rangle \\
\mathit{waitingResponse} = \varnothing \\
\mathit{visiting} = \varnothing \\
\mathit{current} = (\langle\,\rangle, -1) \\
\mathit{next} = (\langle\,\rangle, -1) \\
\hline
\end{array}
$$

5.5.4 MasterAgent Operations

The following eight main operations specify the transitions the MasterAgent class can undergo. They are as below:

5.5.4.1 The *start* Operation

$$\mathit{start} \mathrel{\widehat{=}} \textsc{Init} \fatsemi \mathit{getFirstIdentity}$$

When the object of the MasterAgent is established for first time, the *start* operation initializes all state variables as described in the *INIT* schema, then it will be composed sequentially with the *getFirstIdentity* operation, as will be explained below.

5.5.4.2 The *getFirstIdentity* Operation

$$
\begin{array}{|l}
\multicolumn{1}{l}{\mathit{getFirstIdentity}} \\
\hline
\Delta \mathit{known}, \mathit{waitingResponse}, \mathit{next} \\
\mathit{seed}? : \mathit{Identity} \\
\hline
\mathit{seed}? \notin \operatorname{ran} \mathit{known} \\
\mathit{known}' = \langle \mathit{seed}? \rangle \\
\mathit{waitingResponse}' = \{\mathit{seed}?\} \\
\mathit{next}' = (\langle \mathit{seed}? \rangle, 0) \\
\hline
\end{array}
$$

When the user enters the identity *seed*?, the *mAg* has to add this *seed*? to the sequence *known* which holds all identities whose information the system will retrieve. Also, *seed*? will be added to the *waitingResponse* set to indicate that this identity is assigned to a *gAg* but has not retrieved its information yet. Consequently, the tuple is allocated to *next* to specify that *seed*? profile will be extracted in the next iteration (level 0).

5.5.4.3 The *assignGrabber* Operation

$$
\begin{array}{rl}
\mathit{assignGrabber} \mathrel{\widehat{=}} & [\mathit{first}\ \mathit{next} \neq \langle\,\rangle]\, \mathit{addAssigning} \mathbin{⨟} \mathit{assignGrabber} \\
& \qquad\qquad [] \\
& [\mathit{first}\ \mathit{next} = \langle\,\rangle] \quad \mathit{receiveAllProfiles}
\end{array}
$$

As described in Section 5.3, the first operation the MasterAgent object has to carry out when it receives the *seed*? from the user is to assign it to a *gAg* . The *assignGrabber* is a recursive operation that, when composed with the *addAssigning* operation (as detailed below), will continue until all identities in the sequence in *next* are allocated to *gAg*s.

5.5.4.4 The *addAssigning* Operation

$$
\begin{array}{|l}
\multicolumn{1}{l}{\mathit{addAssigning}} \\
\hline
\Delta \mathit{visiting}, \mathit{next} \\
\mathit{id}! : \mathit{Identity} \\
\hline
\exists\, \mathit{gAg} : \mathit{GrabberAgent} \bullet \mathit{gAg} \notin \operatorname{ran} \mathit{visiting} \Rightarrow \\
\mathit{visiting}' = \mathit{visiting} \cup \{\mathit{gAg} \mapsto \mathit{id}!\} \land \\
\mathit{next}' = (\mathit{tail}\,\mathit{first}\ \mathit{next}, \mathit{second}\ \mathit{next}) \land \\
\mathit{id}! = \mathit{head}(\mathit{first}\ \mathit{next}) \land \\
\mathit{gAg}.\mathit{receiveID} \\
\hline
\end{array}
$$

To express the *addAssigning* operation, assent is given that there exists a *gAg* from the class GrabberAgent, which will be described in Section 5.6, that has not been assigned before to any identity. This *gAg* will be allocated to the head of the sequence of identities in *next* (denoted *id*!).

Consequently, the *id*! will be removed from the sequence and the remaining elements will be shifted to be ready for allocation to the subsequent *gAg* . Note that the iteration level will not increment in this operation. Once *addAssigning* is accomplished successfully, the identity (*id*!) will be passed to the GrabberAgent class for retrieval of its information, as will be explained in Section 5.6.3.1.

5.5.4.5 The *receiveAllProfiles* Operation

$$
\begin{array}{rl}
\mathit{receiveAllProfiles} \mathrel{\widehat=} & [\mathit{waitingResponse} \neq \varnothing]\, \mathit{receiveProfiles} \fatsemi \mathit{receiveAllProfiles} \\
& \qquad\qquad [\,] \\
& \qquad [\mathit{waitingResponse} = \varnothing] \;\; \mathit{completeRetrieving}
\end{array}
$$

The *receiveAllProfiles* operation that was called in *AssignGrabber* operation is also a recursive composition operation, which is composed of the *receiveProfiles* operation. However, it will not move on to the next level of iteration until information about all identities in *waitingResponse* is returned.

5.5.4.6 The *receiveProfiles* Operation

$$\begin{array}{|l}
\multicolumn{1}{l}{\textit{receiveProfiles}} \\
\hline
\Delta known, waitingResponse, next \\
id? : Identity \\
newRec? : Record \\
newFriends? : \operatorname{seq} Identity \\
\hline
waitingResponse \neq \varnothing \\
\textbf{let}\, filtered == (newFriends? \upharpoonright (Identity \setminus \operatorname{ran} known)) \bullet \\
known' = known \frown filtered \\
next' = newFriends? \\
waitingResponse' = waitingResponse \setminus \{id?\} \\
\hline
\end{array}$$

When the *mAg* receives information of a profile (*id*?) from a *gAg* , it has to pass this information (*newRec*? and *newFriends*?) to the LocalRepository class to update the history of the profile *id*?. It is necessary that the *mAg* filters *newFriends*? to remove identities which already exist in *known*. Following this, it must remove this *id*? from the *waitingResponse*.

5.5.4.7 The *completeRetrieving* Operation

$$\begin{array}{rl}
completeRetrieving \mathrel{\widehat{=}} & \neg\, [second\ next = iterationLevel \lor \#known \geq counter] \\
& \qquad\qquad nextLevel \fatsemi assignGrabber \\
& \qquad\qquad\qquad [\,] \\
& [second\ next = iterationLevel \lor \#known \geq counter]\textsc{Init}
\end{array}$$

This operation either applies the *nextLevel* operation recursively or exits and finishes the iteration if one of the stopping criteria is met as described in previous chapter, Section 4.3.2.

5.5.4.8 The *nextLevel* Operation

$$\begin{array}{|l}
\multicolumn{1}{l}{\textit{nextLevel}} \\
\hline
\Delta \mathit{current}, \mathit{next}, \mathit{waitingResponse} \\
\mathit{id}! : \mathrm{seq}\, \mathit{Identity} \\
\hline
\mathit{second}\ \mathit{next} \neq \mathit{iterationLevel} \\
\mathit{counter} \leq \#\mathit{known} \\
\mathit{next}' = (\langle\, \rangle, \mathit{second}\ \mathit{next} + 1) \\
\mathit{current}' = \mathit{next} \\
\mathit{waitingResponse}' = \mathrm{ran}\, \mathit{first}\ \mathit{next} \\
\mathit{mAg}.\mathit{assignGrabber} \\
\hline
\end{array}$$

When the *mAg* has completed retrieving all information about all identities in the *waitingResponse* set, i.e. completes the iteration level, it must move to the next level of iteration. At this stage, all identities in *next* will be copied to both *waitingResponse* and *current* to start a new level of data extraction. The complete MasterAgent class is displayed below in Figure 5.6.

$MasterAgent$

$iterationLevel : \mathbb{N}$

$counter : \mathbb{N}$

$null : Record$

$known : \mathrm{seq}\, Identity$
$waitingResopnse : \mathbb{P}\, Identity$
$visiting : Agent ⤔ Identity$
$current, next : (\mathrm{seq}\, Identity \times \mathbb{Z})$

$waitingResponse \subseteq \mathrm{ran}\, known$
$\mathrm{ran}\, first\ next \subseteq \mathrm{ran}\, known$
$\mathrm{ran}\, first\ current \subseteq \mathrm{ran}\, known$
$\mathrm{disjoint}\langle \mathrm{ran}\, first\ current, \mathrm{ran}\, first\ next \rangle$

$\textsc{Init}$

$known = \langle\,\rangle$
$waitingResponse = \varnothing$
$visiting = \varnothing$
$current = (\langle\,\rangle, -1)$
$next = (\langle\,\rangle, -1)$

$start \mathrel{\widehat=} \textsc{Init} \mathbin{⨾} getFirstIdentity$

$getFirstIdentity$

$\Delta known, waitingResponse, next$
$seed? : Identity$

$seed? \notin \mathrm{ran}\, known$
$known' = \langle seed? \rangle$
$waitingResponse' = \{seed?\}$
$next' = (\langle seed? \rangle, 0)$

$$
\begin{array}{l}
assignGrabber \mathrel{\widehat=} [first\ next \neq \langle\,\rangle]\, addAssigning \mathbin{⨾} assignGrabber \\
\qquad\qquad\qquad\qquad [] \\
\qquad\qquad\quad [first\ next = \langle\,\rangle] \quad receiveAllProfiles
\end{array}
$$

$MasterAgent(cont)$

$addAssigning$

$\Delta visiting, next$
$id! : Identity$

$\exists\, gAg : GrabberAgent \bullet gAg \notin \operatorname{ran} visiting \Rightarrow$
$visiting' = visiting \cup \{gAg \mapsto id!\} \wedge$
$next' = (tail\, first\ next, second\ next) \wedge$
$id! = head(first\ next) \wedge$
$gAg.receiveID$

$$
\begin{aligned}
&receiveAllProfiles \mathrel{\widehat=}\\
&\qquad [waitingResponse \neq \varnothing] receiveProfiles \,\fatsemi\, receiveAllProfiles\\
&\qquad\qquad\qquad\qquad [\,]\\
&\qquad\qquad\qquad [waitingResponse = \varnothing] completeRetrieving
\end{aligned}
$$

$receiveProfiles$

$\Delta known, waitingResponse, next$
$id? : Identity$
$newRec? : Record$
$newFriends? : \operatorname{seq} Identity$

$waitingResponse \neq \varnothing$
$\mathbf{let}\, filtered == (newFriends? \upharpoonright (Identity \setminus \operatorname{ran} known)) \bullet$
$known' = known \frown filtered$
$next' = newFriends?$
$waitingResponse' = waitingResponse \setminus \{id?\}$

$$
\begin{aligned}
&completeRetrieving \mathrel{\widehat=}\\
&\qquad \neg\, [second\ next = iterationLevel \vee \#known \geq counter]\\
&\qquad\qquad nextLevel \,\fatsemi\, assignGrabber\\
&\qquad\qquad\qquad\qquad [\,]\\
&\qquad\ [second\ next = iterationLevel \vee \#known \geq counter]\textsc{Init}
\end{aligned}
$$

$nextLevel$

$\Delta current, next, waitingResponse$
$id! : \operatorname{seq} Identity$

$second\ next \neq iterationLevel$
$counter \leq \#known$
$next' = (\langle\,\rangle, second\ next + 1) \wedge$
$current' = next \wedge$
$waitingResponse' = \operatorname{ran} first\ next \wedge$
$mAg.assignGrabber$

Figure 5.6: Formal Description of MasterAgent Class

5.6 GrabberAgent Class

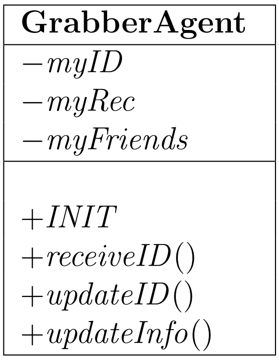


Figure 5.7: GrabberAgent Class Diagram

The second main class in the OSNRS is the GrabberAgent. The objects of the GrabberAgent class in the OSNRS could be considered as the worker bees in the bees' colony. This is because the GrabberAgent objects must visit the OSN profile, extract data, monitor updates and send regular reports to the MasterAgent class. Figure 5.7 shows the class diagram of the GrabberAgent, in which its components are detailed.

5.6.1 State Schema of GrabberAgent

StateSchema

$myID : Identity$
$myRec : Record$
$myFriends : \mathrm{seq}\, Identity$

$myID \notin \mathrm{ran}\, myFriends$

The state schema for the GrabberAgent class contains three state variables:

1. *myID*: denotes the unique profile identity that the *gAg* is assigned to.
2. *myRec*: denotes the personal information of the profile identity.
3. *myFriends*: denotes the list of friends of the profile identity.

The predicate of the state schema is true if and only if each identity is not a friend of itself.

5.6.2 The Initial State

$$\begin{array}{|l}
\textit{INIT} \\
\hline
\textit{myRec} = \textit{null} \\
\textit{myFriends} = \langle\,\rangle \\
\hline
\end{array}$$

When an object of the class GrabberAgent is created, it has not yet been allocated to any identity. Accordingly, neither the personal information of the identity nor its list of friends are known. Thus, the state variable *myRec* is initialized to the constant *null* while *myFriends* is set to an empty sequence.

5.6.3 GrabberAgent Operations

Although GrabberAgent has only three main operations, these are what the OSNRS is based upon. They are as follows:

5.6.3.1 The *receiveID* Operation

$$\textit{receiveID} \mathrel{\widehat{=}} \textsc{Init} \fatsemi \textit{updateID} \fatsemi \textit{updateInfo}$$

As mentioned earlier in the *addAssigning* operation of MasterAgent class (Section 5.5.4.4), when the *mAg* assigns a *gAg* to an identity, the GrabberAgent class will get this identity through a *receiveID* operation. The *receiveID* has to compose several operations sequentially. If the *id*? is received for first time, then the *INIT* will be executed as described above. Otherwise, i.e. where the *id*? has been received before, one of the following operations will be applied; the *updateInfo* operation or the *updateID* operation.

5.6.3.2 The *updateInfo* Operation

$$
\begin{array}{|l}
\hline
\textit{updateInfo} \\
\textit{rec}! : \textit{Record} \\
\textit{friends}! : \mathrm{seq}\, \textit{Identity} \\
\textit{id}! : \textit{Identity} \\
\hline
\exists\, \textit{newRec}? : \textit{Record}, \textit{newFriends}? : \mathrm{seq}\, \textit{Identity} \bullet \\
\textit{newRec}? \neq \textit{myRec} \lor \textit{newFriends}? \neq \textit{myFriends} \Rightarrow \\
\textit{myRec}' = \textit{newRec}? \land \\
\textit{myFriends}' = \textit{newFriends}? \land \\
\textit{rec}! = \textit{myRec}' \land \\
\textit{friends}! = \textit{myFriends}` \land \\
\textit{id}! = \textit{myID} \land \\
\textit{mAg.receiveProfile} \\
\hline
\end{array}
$$

The *updateInfo* operation is invoked periodically after the *gAg* has been assigned an id?. Since the retrieved information is coming from an external database (OSN server), we abstract away from the querying process by assuming that there exists a record (*newRec*?) that contains all personal information about *myID*, and a sequence of identities (*newFriends*?) that holds a list of *myID* friends. Either the *newRec*? or *newFriends*?, or both, should be distinguished from the existing *myRec* and *myFriends* respectively in order to be replaced. As a result, the new information in *myRec*, *myFriends* accompanied with *myID* will be sent back to the MasterAgent class through *rec*!, *friends*! and *ID*! consecutively when the *mAg.receiveProfile* operation is involved.

5.6.3.3 The *updateID* Operation

The *upadateID* operation is called upon to replace the old identity that the *gAg* is allocated to with a new identity (*id*?). This is important in order to re-use the *gAg* in cases where the identity that it is assigned to has been removed from the sub network of OSN, or a *gAg* is assigned to a new profile.

$$
\begin{array}{|l}
\textit{updateID} \\
\hline
\Delta \mathit{myID} \\
\mathit{id}? : \mathit{Identity} \\
\hline
\mathit{myID}' = \mathit{id}? \\
\hline
\end{array}
$$

The Object-Z specification of the class GrabberAgent is given in Figure 5.8

$$
\begin{array}{|l}
\textit{GrabberAgent} \\
\hline
\begin{array}{|l}
\hline
\mathit{myID} : \mathit{Identity} \\
\mathit{myRec} : \mathit{Record} \\
\mathit{myFriends} : \operatorname{seq} \mathit{Identity} \\
\hline
\mathit{myID} \notin \operatorname{ran} \mathit{myFriends} \\
\hline
\end{array}
\quad
\begin{array}{|l}
\textsc{Init} \\
\hline
\mathit{myRec} = \mathit{null} \\
\mathit{myFriends} = \langle\, \rangle \\
\hline
\end{array} \\
\\
\begin{array}{|l}
\textit{updateInfo} \\
\hline
\mathit{rec}! : \mathit{Record} \\
\mathit{friends}! : \operatorname{seq} \mathit{Identity} \\
\mathit{id}! : \mathit{Identity} \\
\hline
\exists\, \mathit{newRec}? : \mathit{Record}, \mathit{newFriends}? : \operatorname{seq} \mathit{Identity} \bullet \\
\mathit{newRec}? \neq \mathit{myRec} \lor \mathit{newFriends}? \neq \mathit{myFriends} \Rightarrow \\
\mathit{myRec}' = \mathit{newRec}? \land \\
\mathit{myFriends}' = \mathit{newFriends}? \land \\
\mathit{rec}! = \mathit{myRec}' \land \\
\mathit{friends}! = \mathit{myFriends}' \land \\
\mathit{id}! = \mathit{myID} \land \\
\mathit{mAg.receiveProfile} \\
\hline
\end{array} \\
\mathit{receiveID} \mathrel{\widehat=} \textsc{Init} \mathbin{⨾} \mathit{updateID} \mathbin{⨾} \mathit{updateInfo} \\
\begin{array}{|l}
\textit{updateID} \\
\hline
\Delta \mathit{myID} \\
\mathit{id}? : \mathit{Identity} \\
\hline
\mathit{myID}' = \mathit{id}? \\
\hline
\end{array} \\
\hline
\end{array}
$$

Figure 5.8: Formal Description of GrabberAgent Class

5.7 Summary

This chapter has presented, through Object-Z specification, the feasibility of using MAS technology in extracting historical information from OSN sites. The aim was to ensure that the proposed system (OSNRS) is robust, reliable and fits its purpose, before the implementation phase. Object-Z is very close to being an executable specification because it has the ability to be converted into computer code easily. Next chapter will describe the implementation of OSNRS.

Chapter 6

Design and Implementation of a Multi Agent System for Historical Data Extraction from Online Social Networks

Preamble

The desire to track information changes in real time requires an adaptation to previous methods in web DE. In the previous chapter, a formal specification of an OSN system was constructed to help us investigate the feasibility of using MAS technology in retrieving historical information from OSN sites.

This chapter applies MAS technology in retrieving information from OSNs. An algorithm making use of MAS within the OSNRS is proposed. The novelty of the proposed approach is in associating an agent in each user profile to monitor its updates, which are sent to a controller agent that saves a history of each users activity in a local repository.

6.1 Introduction

The increased simplicity in accessing the WWW via wireless devices such as laptops and smart phones helps end users to participate in OSNs. OSN gives them more opportunity to make new friendships, share their interests even with unknown people, upload photos and distribute their personal information over time. These changes in the users' profiles may affect the behaviour and actions of their friends and lead to alterations in the network analysis. Thus, more attention is required as to how information will be collected from OSN websites taking into consideration these rapid changes.

This chapter focuses on tracking profiles changes and extracting historical information from OSN sites. Although there are numerous studies which have attempted to extract millions of profiles from different OSNs, e.g. [45, 102], to date, they have analyzed results gained by visiting each profile only once. To the best of the current authors' knowledge, there is no reported work which deals with monitoring the rapid changes in those profiles.

In Chapter 4, a parser was developed for the automated extraction of personal information of OSN profiles and their list of friends based on the BFS algorithm. However, that approach has several limitations which need to be addressed regarding using a centralized system and a parser (string tokenizer) to retrieve information from the profiles' source pages. Moreover, the previous algorithm did not address the monitoring of profiles over time.

Thus, the aim of this chapter is to improve the algorithm presented in Chapter 4 through use of MAS to overcome some of the limitations in preserving and recording the temporal ordering of OSN profiles' events, to expedite the extraction process and to make the algorithm more scalable.

The remainder of the chapter is arranged as follows: Section 6.2 presents the conceptual overview of the contributions to OSNRS. Section 6.3 details the implementation of running agents on MySpace profiles, while findings and results are analyzed in Section 6.4. Section 6.5 validates the system and the limitations of current approach is illustrated in Section 6.6. Finally, the summary is presented in Section 6.7.

6.2 Conceptual Overview of Online Social Network Retrieval System

The developed Online Social Network Retrieval System (OSNRS) aims to overcome one of the drawbacks of the Social Networking Extractor whereby the application is stopped and terminates if it fails to parse one of the profiles. This issue could be avoided if the information of the OSN profiles is extracted in a multi-threading approach rather than using a single thread. As demonstrated in Chapter 5 through formal specification, MAS is a useful technology to fulfil these aims.

6.2.1 Selecting the software

Since the OSNRS relies mainly on creating a MAS, choosing the appropriate programming language and development tools are a significant issue in the success of building a MAS [30](p.8). Although major programming languages such as Java and C++ could be used to create the MAS, using tools that provide a relatively straightforward implementation of MAS concepts would be helpful and time saving.

In building the OSNRS, the search for tools was restricted to those built in Java, in order to ensure compatibility with the parser developed in Chapter 4. Three different Agent tools have been evaluated to build the OSNRS. The common issue with using all these tools is the incompatibility between their requirements and available facilities, due to these tools advancing slowly with the development of operating systems and other requirements. Besides, the attached manuals are not up-to-date. The tools are:

1. **AgentBuilder**: is the first tool selected to build the agents. At the time

of starting to code the agents in 2007, the latest Microsoft Windows desktop operating system was Vista, while the AgentBuilder was designed to work with Windows XP.

There were compatibility issues in using this tool. Moreover, AgentBuilder is a commercial tool, i.e. not free software, and thus it was difficult to find solutions for errors. The only option was to contact the developers, which proved to be seriously time-consuming.

2. **Aglets**: is an open source platform and library for developing an agent in Java. Many universities use it to support their courses. Appendix (B) presents a brief overview of Aglets. The difficulty encountered with this tool was that its latest version was released in 2004, and its forum was therefore relatively inactive and inefficient.

3. **JADE**: stands for Java Agent DEvelopment framework. (JADE) has been selected to implement OSNRS because it is a widely adopted open source software tools for designing MAS in recent years. JADE is a " *distributed middleware system with a flexible infrastructure allowing easy extension with add-on modules*" [30] (p.18).

   As JADE is written in Java, this encouraged its selection, especially as it fits well with the developed parser, which is described in Chapter 4. Appendix (C) provides more information on JADE and how to use it.

6.2.2 Organizational Model of OSNRS

The Organizational model of the OSNRS is built based on the structure of MAS in general and the selected tool in particular.

JADE is selected, as with Camacho et al. in [43] and Aldea et al. in [18], to implement the OSNRS. In JADE, each agent should live in a running instance of

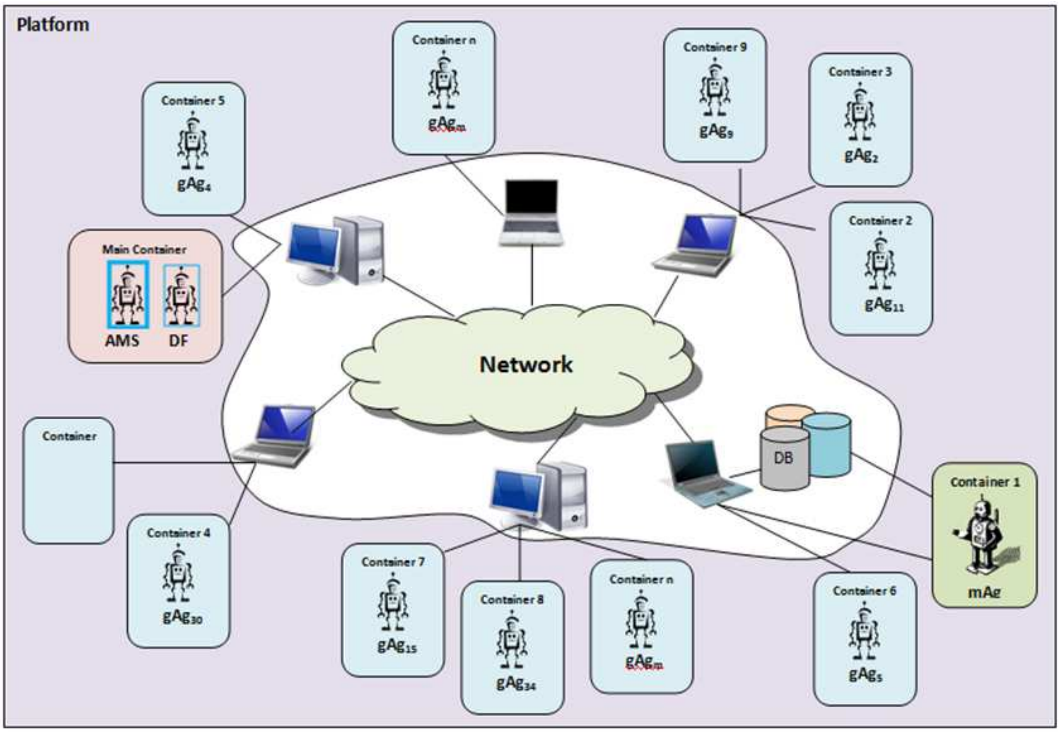


Figure 6.1: Organizational Model of the Proposed System

runtime environment called a container. A composite of containers comprises the platform.

Although many agents may live in one container, and several containers compose a platforms, and the system is composed of one or more platform. it was decided to simplify the OSNRS environment by:

- Assigning one platform to compose all containers.
- Creating each agent in a unique container.

Figure 6.1 shows the organizational model of OSNRS and its environment. Note that the number of containers in the workstations can vary from 0 to many. Also, the local repository (Database) can be accessed only by the *mAg* .

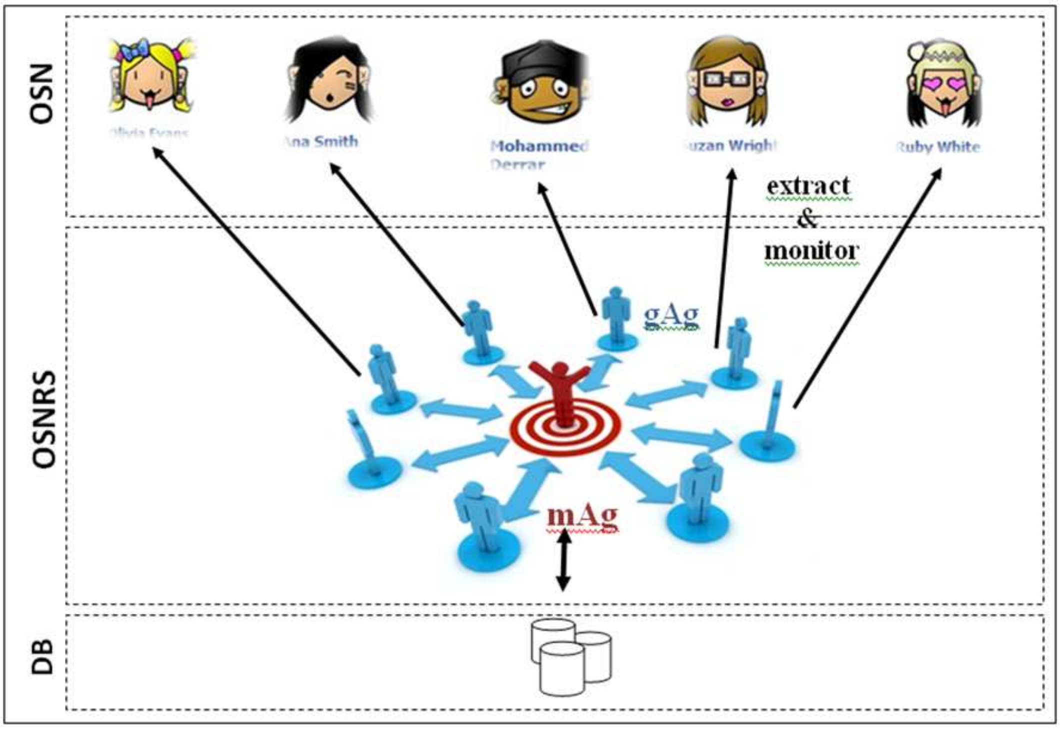


Figure 6.2: Structural Model of OSNRS (Repeated)

6.2.3 Structural Model of OSNRS

In Section 5.2, the OSNRS was defined as a MAS that consists of a finite set of agents (grabber agents) controlled by a special agent called a MasterAgent. Each grabber agent (*gAg*) will be allocated to a unique URL to extract the information from an OSN profile and will continue to monitor any updates in the profile. The extracted information will be sent back to the *mAg* , which creates a history of each profile, recorded in a local repository to be ready for further analysis. As a reminder, Figure 6.2 which represents the structural model of OSNRS is repeated from the previous chapter (Figure 5.1 in Section 5.2).

The novelty of the OSNRS is in keeping the *gAg* monitoring the assigned profile for updates. Such updates are detected by comparing the current retrieved information with the saved file from a previous extraction. Once a change is captured, the *gAg* will inform the *mAg* and send a copy of the updated file.

Several MAS features fit well with the OSNRS. The proactivity feature of mAg means that it initiates achievement of the goal for which it was designed, which as stated is to collect the historical information of the seed profile and that of its friends. Thus, it will start to behave as a stand alone process (autonomy feature) through taking decisions such as:

- how to find the existing *gAg* in the platforms?
- which *gAg* should be allocated to a URL?
- whether the URL should be added to or removed from the queue.
- when it should terminate the application?
- when it should reactivate a *gAg* ? i.e. deciding whether a specified time is appropriate to the extraction process.

The sociability feature of *mAg* allows it to communicate with the *gAg*s, the user, and other software e.g. database systems. *mAg* communicates with *gAg*s using messages to exchange plans and goals. For instance, the *mAg* must send a message to a *gAg* to:

- allocate the *gAg* to a URL.
- ask the *gAg* to stop extracting information from a profile.
- change the targeted profile.
- reactivate the *gAg* to start monitoring.

The *mAg* perceptivity feature comes into play when it balances between the directed goal (to extract historical information from OSN profiles) and the timely response to factors detected in the environment. For example, if a *gAg* reports that the allocated URL is broken or unavailable, the *mAg* must take a decision to remove the URL from the queue and the database as well as releasing the *gAg* to make it free for allocation to another URL (if necessary).

On the other hand, MAS features show up in the *gAg* as follows; the autonomy of a *gAg* allows it to operate as a standalone process to achieve its goal (retrieving the profile information) without the direct intervention of users. The perceptivity of the *gAg* permits it to detect changes in profiles and to make decisions as to when it must report the results to the *mAg* .

The mobility feature permits the *gAg*s to be distributed between different machines in the network. The distribution of *gAg*s helps to speed up the operation of extraction. The scalability of the MAS will facilitate the process of adding new *gAg* or other agents for different purposes as required without having to change the existing system. The sociability allows *gAg* , *mAg* and users of OSNRS to communicate with each other and exchange their knowledge.

6.2.4 Flow of Information between OSNRS Components

By applying MAS technology to the OSNRS, the tasks of the previous application "Social Networking Extractor" will be modified and split between *mAg* and *gAg* .

Figure 6.3 shows the sequence of main actions between the OSNRS' major classes. When the object *mAg* of MasterAgent class is created successfully in the container, it will prompt the user to specify the time to reactivate the *gAg* , in addition to the seed URL and the stopping criteria. The *mAg* has to validate the correctness of the URL as described earlier in Section 4.2.2. This seed URL will be added to the front of the queue. Also the *mAg* must check the stopping criteria before starting the process of extraction.

The extraction process starts when the *mAg* looks at the platform for a *gAg* , to allocate the profile URL to it. Allocation requires ensuring that the *gAg* has not been allocated to any profile before, nor that the URL is duplicated. In other words, each profile should be parsed by just one *gAg* . If no free *gAg* is found, the *mAg* must create a *gAg* in a new container and add it to the platform.

Agents use messages to communicate with each other. JADE agents use Agent

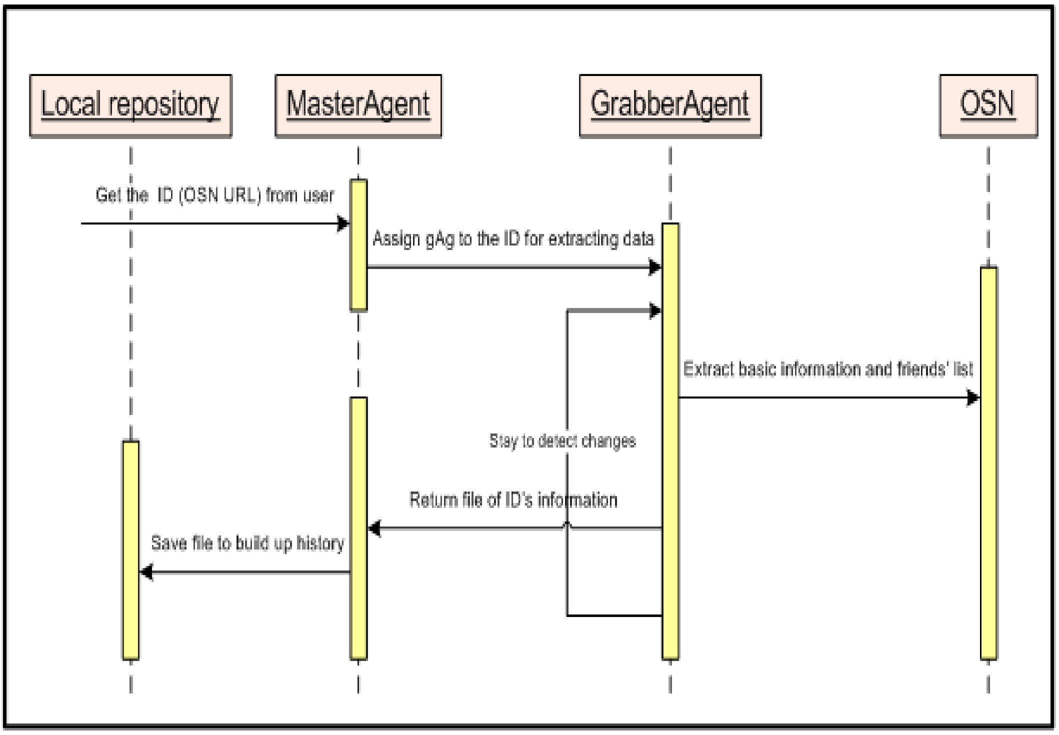

Local repository
MasterAgent
GrabberAgent
OSN
Get the ID (OSN URL) from user
Assign gAg to the ID for extracting data
Extract basic information and friends' list
Stay to detect changes
Return file of ID's information
Save file to build up history

Figure 6.3: Sequence Diagram of OSNRS (Repeated)

Communication Language (ACL) which follows FIPA ACL[1] standard; the most used and studied agent communication language for messaging format. The most common message attributes include:

- The sender (initialized automatically).
- List of receiver(s).
- Performative act: the goal of sending the message, e.g. to inform, request, query, etc.
- Content of the message.
- Conversation_ID: to link messages in the same conversation.

More details about JADE messaging can be found in Appendix (C).

Referring back to Figure 6.3, the *gAg* will receive the message and accept the task. It will then start to extract profile information after ensuring the availability

[1] http://www.fipa.org/specs/fipa00061/

of the profile. In addition to the basic information, the list of friends which are hyperlinked to the users' profiles is extracted.

A copy of this information will be saved in a file and another copy will be sent back to the *mAg* using a different port. To speed up the extraction process, a port is set up just for transferring files to distinguish it from the port that is used by agents for other communications with the *mAg* . After submitting the file, the *gAg* will go to sleep until the *mAg* decides to reactivate it in line with the time period specified by the user. This time could be seconds, minutes, hours or even days according to the purpose of extraction and the required information. When the *gAg* is reactivated, it will repeat the process of extraction, but when it gets the file, it must now compare the newly collected data with the previously saved data. If updates are captured, the *gAg* must replace the saved file and inform the *mAg* of the new updates (if nothing is found, the *gAg* will not send anything to the *mAg*).

The *mAg* saves each returned file in a local repository. The repository is organized in such a way that the updated files, which are each related to the same profile, are linked together to form a history of each profile. Also, each URL in the friends list will be added to the rear of the queue after checking that it has not been added before. This queue is used to ensure that the URL is allocated and parsed by only one *gAg* . By completing this stage, the first iteration is accomplished.

The *mAg* works sequentially in terms of behaviour to deal with all the tasks explained above. In order to allocate URLs in the friends list queue to *gAg*s, *mAg* applies one of the features of MAS which allows it to carry out tasks in parallel using a composition of behaviours, including parallel behaviour and cyclic behaviour in addition to sequential behaviour. Figure 6.4 shows the classification of behaviours which JADE provides for its agents. Appendix (C) presents some examples of these behaviours.

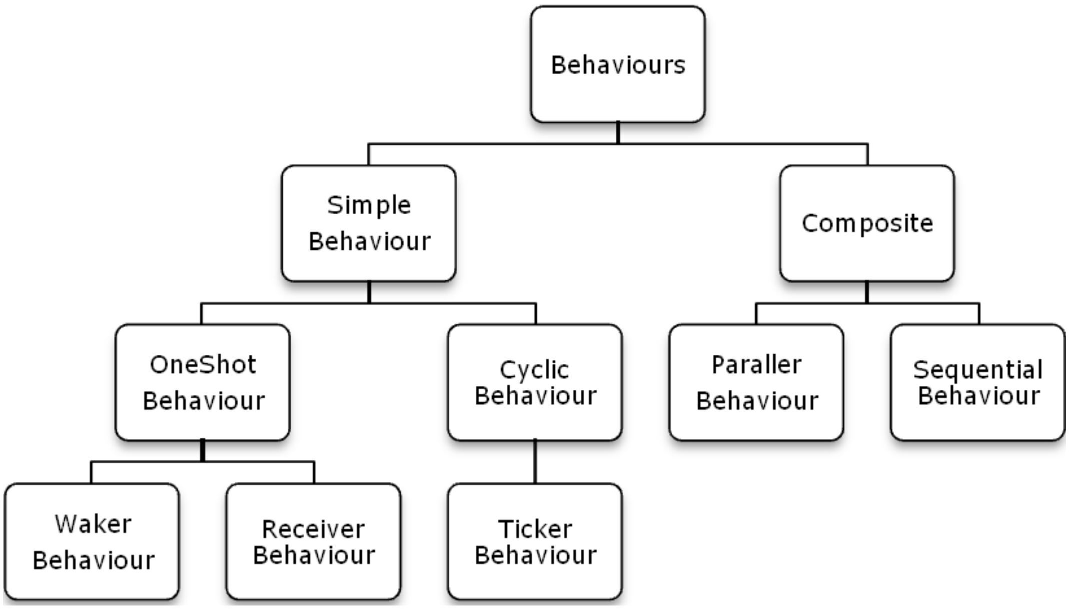


Figure 6.4: Classification of JADE Agent Behaviour

When each *gAg* in the queue is allocated to a URL, it will repeat what the first *gAg* does as explained above. Note that the *mAg* will check the stopping criteria to decide when to stop allocating *gAg*s.

6.3 Implementation of Online Social Network Retrieval System

This section highlights the steps required to implement the proposed system (OS-NRS) taking into consideration the aim of developing an algorithm which is able to overcome the limitations in the previous approach which was presented in Chapter 4, by applying MAS technology. The flowchart in Figure 6.5 details the steps of the OSNRS experiment.

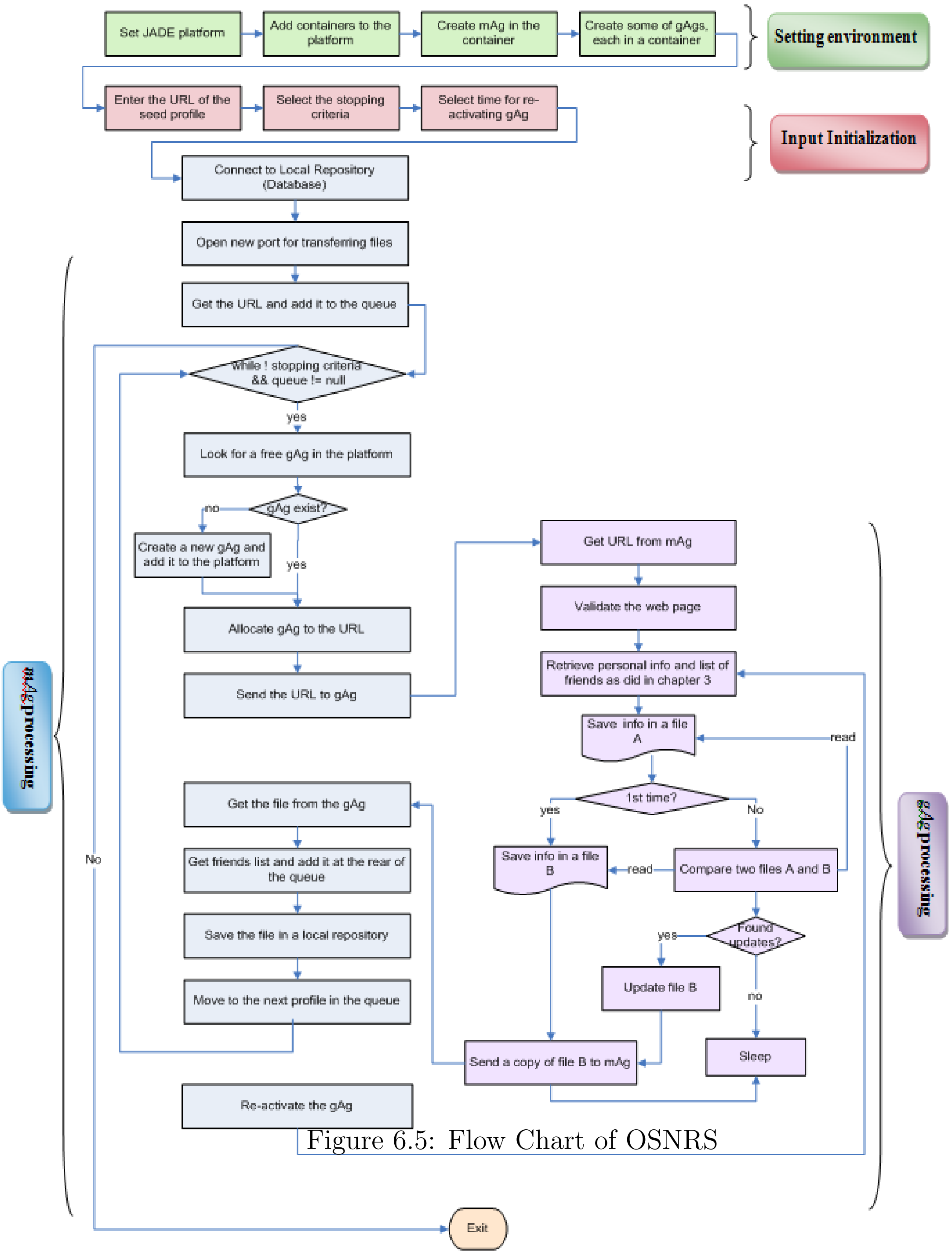
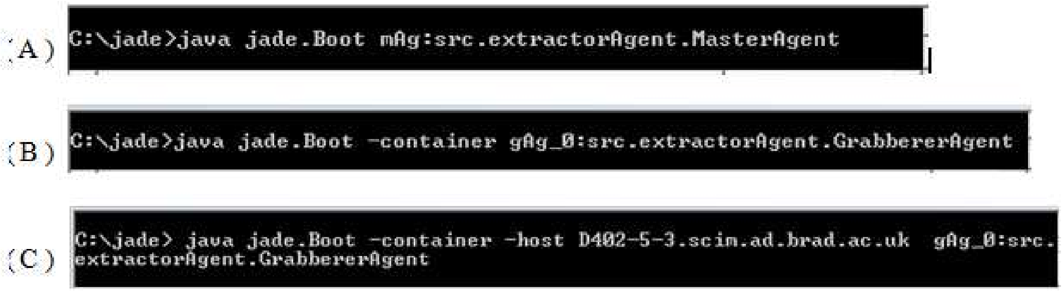

Set JADE platform
Add containers to the platform
Create mAg in the container
Create some of gAgs, each in a container
Setting environment
Enter the URL of the seed profile
Select the stopping criteria
Select time for re-activating gAg
Input Initialization
Connect to Local Repository (Database)
Open new port for transferring files
Get the URL and add it to the queue
While ! stopping criteria && queue != null
yes
Look for a free gAg in the platform
gAg exist?
no
Create a new gAg and add it to the platform
yes
Allocate gAg to the URL
Send the URL to gAg
Get the file from the gAg
Get friends list and add it at the rear of the queue
Save the file in a local repository
Move to the next profile in the queue
Re-activate the gAg
No
Exit
mAg processing
Get URL from mAg
Validate the web page
Retrieve personal info and list of friends as did in chapter 3
Save info in a file A
read
1st time?
yes
No
Save info in a file B
read
Compare two files A and B
Found updates?
yes
no
Update file B
Send a copy of file B to mAg
Sleep
gAg processing

Figure 6.5: Flow Chart of OSNRS



```
C:\jade>java jade.Boot mAg:src.extractorAgent.MasterAgent
```

```
C:\jade>java jade.Boot -container gAg_0:src.extractorAgent.GrabbererAgent
```

```
C:\jade> java jade.Boot -container -host D402-5-3.scim.ad.brad.ac.uk  gAg_0:src.
extractorAgent.GrabbererAgent
```

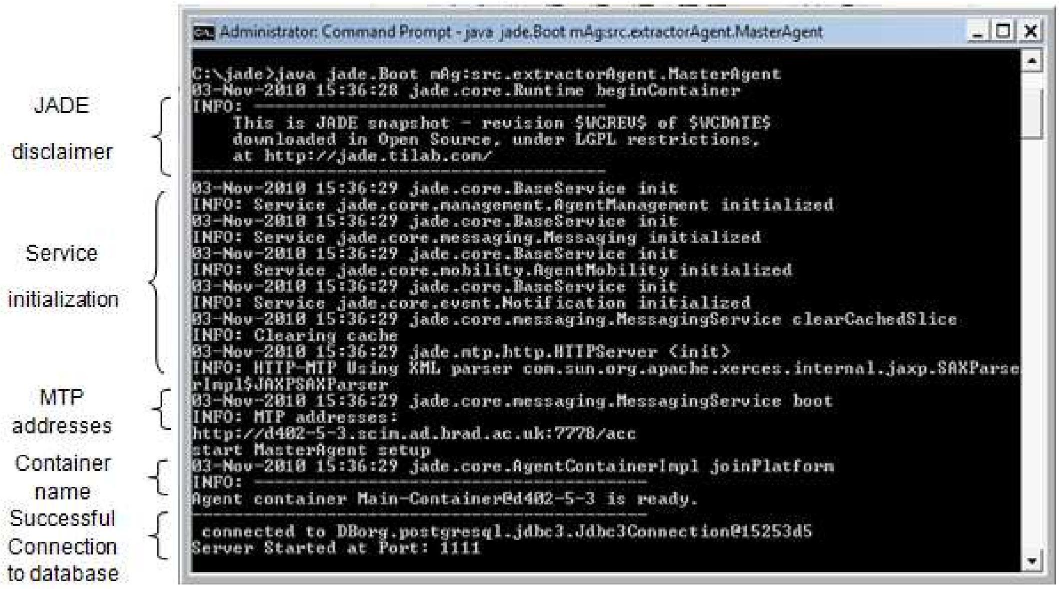
Figure 6.6: Setting JADE Environment for OSNRS

The implementation of the work is accomplished by the following:

6.3.1 Setting the Environment of OSNRS

Setting the environment involves using two connected workstations to share one JADE platform.

The containers are distributed on these two workstations to form the runtime environment. Initially, the *mAg* is created to live in a container in a machine (e.g. called D402-5-3) as an object of the class MasterAgent by the command in Figure 6.6(A), where *mAg* is the nickname of the agent. The successful creation of the *mAg* gives what is shown in Figure 6.7.


Administrator: Command Prompt - java jade.Boot mAg:src.extractorAgent.MasterAgent
JADE disclaimer
Service initialization
MTP addresses
Container name
Successful Connection to database

```
C:\jade>java jade.Boot mAg:src.extractorAgent.MasterAgent
03-Nov-2010 15:36:28 jade.core.Runtime beginContainer
INFO: ----------------------------------
    This is JADE snapshot - revision $WCREV$ of $WCDATE$
    downloaded in Open Source, under LGPL restrictions,
    at http://jade.tilab.com/
----------------------------------------
03-Nov-2010 15:36:29 jade.core.BaseService init
INFO: Service jade.core.management.AgentManagement initialized
03-Nov-2010 15:36:29 jade.core.BaseService init
INFO: Service jade.core.messaging.Messaging initialized
03-Nov-2010 15:36:29 jade.core.BaseService init
INFO: Service jade.core.mobility.AgentMobility initialized
03-Nov-2010 15:36:29 jade.core.BaseService init
INFO: Service jade.core.event.Notification initialized
03-Nov-2010 15:36:29 jade.core.messaging.MessagingService clearCachedSlice
INFO: Clearing cache
03-Nov-2010 15:36:29 jade.mtp.http.HTTPServer <init>
INFO: HTTP-MTP Using XML parser com.sun.org.apache.xerces.internal.jaxp.SAXParse
rImpl$JAXPSAXParser
03-Nov-2010 15:36:29 jade.core.messaging.MessagingService boot
INFO: MTP addresses:
http://d402-5-3.scim.ad.brad.ac.uk:7778/acc
start MasterAgent setup
03-Nov-2010 15:36:29 jade.core.AgentContainerImpl joinPlatform
INFO: --------------------------------------
Agent container Main-Container@d402-5-3 is ready.
--------------------------------------------
 connected to DBorg.postgresql.jdbc3.Jdbc3Connection@15253d5
Server Started at Port: 1111
```

Figure 6.7: Creation of MasterAgent

For creating *gAg*(s), if the *gAg* is created on the same *mAg* workstation, then the *gAg* container should be attached to the main container (*mAg* container). If the *gAg* is created on another workstation, then the host of the main container should be added as shown in Figures 6.6(B) and 6.6(C) respectively.

6.3.2 Input Initialization

The *mAg* will prompt the user to specify 3 variables:

- *The URL of the seed profile*: the *mAg* must validate the correctness of the URL as described earlier in Section 4.2.2.
- *The stopping criteria*: although the experiment could be stopped by one of stopping criteria explained in Section 4.3.2, the stopping of the OSNRS in the experiment is limited only if the second condition is met; i.e. if the friends of the friends of the seed profile are reached. Limiting the stopping criteria to this aims to simplify the extraction process and reduce the time required to complete it. This is because monitoring the updates to profile information requires time almost equivalent to extracting new friend information.
- *The time to reactivate the gAg* : as mentioned previously, specifying reactivation time depends on the attributes of interest in the profile, which differ according the purpose of extraction. At this stage of the experiment, the aim was set to monitor how often the user changes his list of friends (either top friends or all friends) in order to help calculate vulnerability over time [8]. Adding and/or removing friends is not subject to rapid changes; therefore it was decided that the *gAg* should be reactivated (awakened) once a day for two weeks.

6.3.3 Choosing the Sample of OSNRS

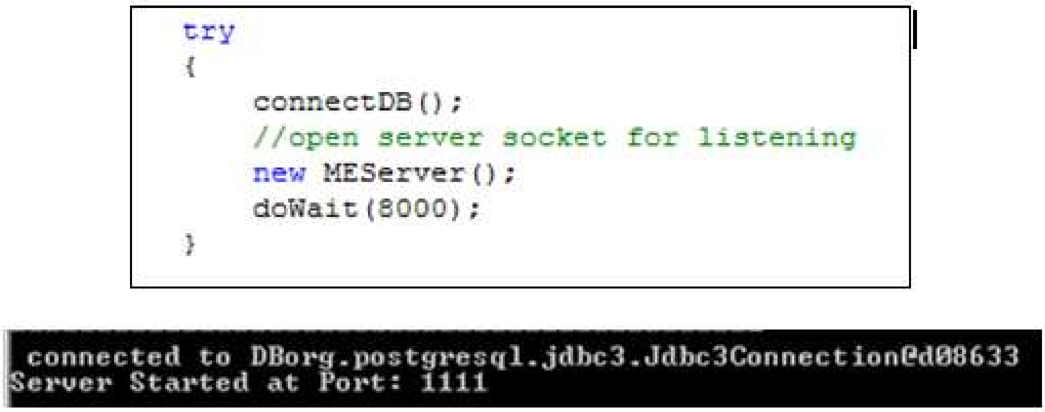
The experiment is a continuation of the work of the previous application, where Myspace profiles form the domain of extraction. The selected sample has been divided into two groups as follows: for group 1, 20 random public profiles are set to be the seed URLs. There are no connections between these profiles, i.e. no profile is in the list of top friends of other profiles.

In contrast, group 2 is formed through choosing the profile which has the largest number of top friends in group 1 (in this case 35 top friends). Subsequently, each public profile in this list was established as a seed URL for extraction. As a result, it was ensured that there are connections between these profiles. Choosing two groups may help in assessing whether the behaviour of the users of the sub network affect each other if they are in connected or disconnected groups.

6.3.4 Running the OSNRS

Once an object of the MasterAgent class is created successfully, it will establish a connection with the local repository. As previously, Postgresql has been chosen to save the retrieved information in rational database tables.

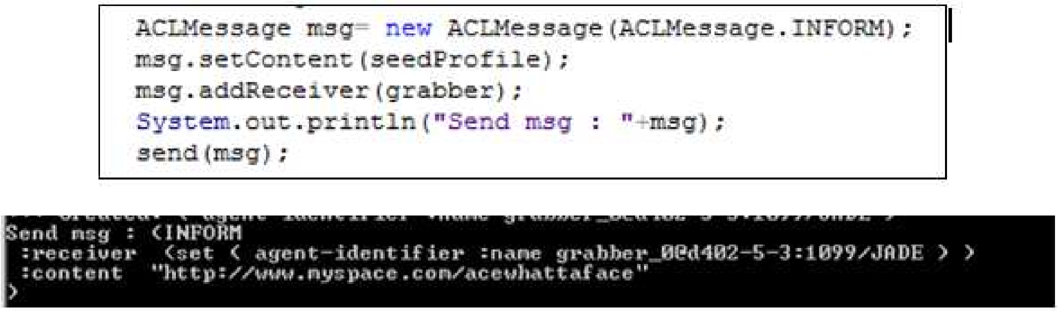
Note that the *mAg* uses the method `dowait (time_in_millhseconds)` to move



```
try
{
    connectDB();
    //open server socket for listening
    new MEServer();
    doWait(8000);
}
```

```
connected to DBorg.postgresql.jdbc3.Jdbc3Connection@d08633
Server Started at Port: 1111
```

Figure 6.8: *mAg* Connection to Database



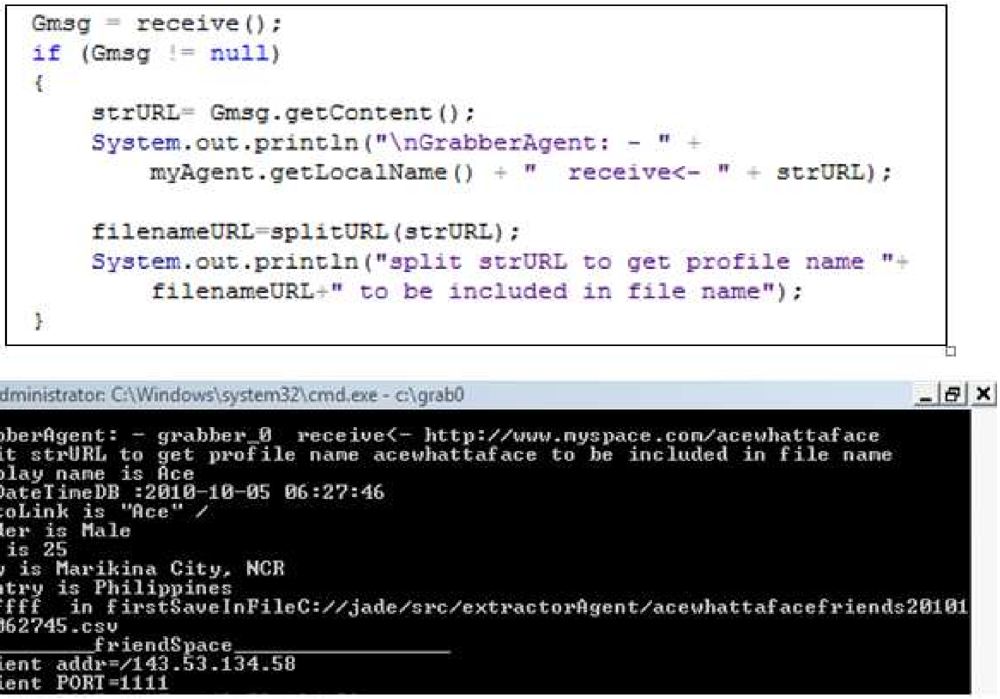
```
ACLMessage msg= new ACLMessage(ACLMessage.INFORM);
msg.setContent(seedProfile);
msg.addReceiver(grabber);
System.out.println("Send msg : "+msg);
send(msg);
```

```
Send msg : (INFORM
 :receiver  (set ( agent-identifier :name grabber_0@d402-5-3:1099/JADE ) )
 :content  "http://www.myspace.com/acewhattaface"
)
```

Figure 6.9: Message for Allocating *gAg* to a URL

from an active state to a waiting state in order to allow these settings to be established, as shown in Figure 6.8. After the specified time has passed, the *mAg* gets the seed URL and adds it into the front of the friends list queue.

It then begins the loop for extraction, starting by looking for a free *gAg* , if any exists, to allocate to the seed URL. Otherwise, the *mAg* will create one and add it to the platform. Allocation of the URL to a *gAg* is accomplished by sending a message (see Figure 6.9).



```
Gmsg = receive();
if (Gmsg != null)
{
    strURL= Gmsg.getContent();
    System.out.println("\nGrabberAgent: - " +
        myAgent.getLocalName() + "  receive<- " + strURL);

    filenameURL=splitURL(strURL);
    System.out.println("split strURL to get profile name "+
        filenameURL+" to be included in file name");
}
```

```
GrabberAgent: - grabber_0  receive<- http://www.myspace.com/acewhattaface
split strURL to get profile name acewhattaface to be included in file name
Display name is Ace
getDateTimeDB :2010-10-05 06:27:46
photoLink is "Ace" /
gender is Male
age is 25
city is Marikina City, NCR
country is Philippines
fffffff  in firstSaveInFileC://jade/src/extractorAgent/acewhattafacefriends20101
005062745.csv
__________friendSpace____________________
FClient addr=/143.53.134.58
FClient PORT=1111
```

Figure 6.10: *gAg* Receiving Message

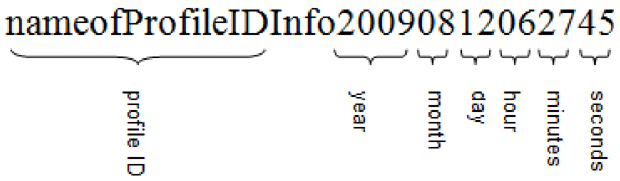


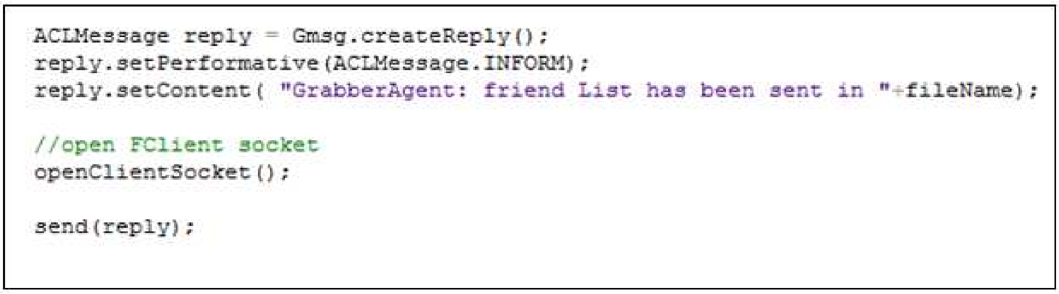
Figure 6.11: File Name Format

Subsequently, the receiver agent (*gAg*) will pick up the message from its mailbox by using the `receive ( )` method as shown in Figure 6.10. Once the *gAg* reads the content of the message, which contains the URL, it will validate the web page to be browsed.

As previously described in Chapter 4, validation includes checking that the profile is not related to a musician, magazine or band. These types of profiles have been excluded because they do not reflect a relationship between individuals. Also, validation involves checking if the web page is broken, has been changed from public to private or has been removed. The *gAg* reports to the *mAg* if one of these situations has occurred. Otherwise, the *gAg* retrieves the list of top friends and number of all friends, in addition to the personal information, using the parser that is described in Chapter 4.

As soon as the *gAg* has completed extracting data, it will store it in a file. The file name is specified at the time of creation in the following format:

Profile unique ID (either name or ID number) + "Info" + current time in (YYYYMMDDhhmmss) format, where 'Y' used for year, 'M' for month, 'D' for day, 'h' for hour, 'm' for minute and 's' for second. For example, a similar file name in Figure 6.10 is shown in Figure 6.11. Such a format guarantees that the file is unique and simplifies the saving of a history of each profile. The file is attached with a reply message to the sender (*mAg*) but on another port as described previously in Section 6.2.4.

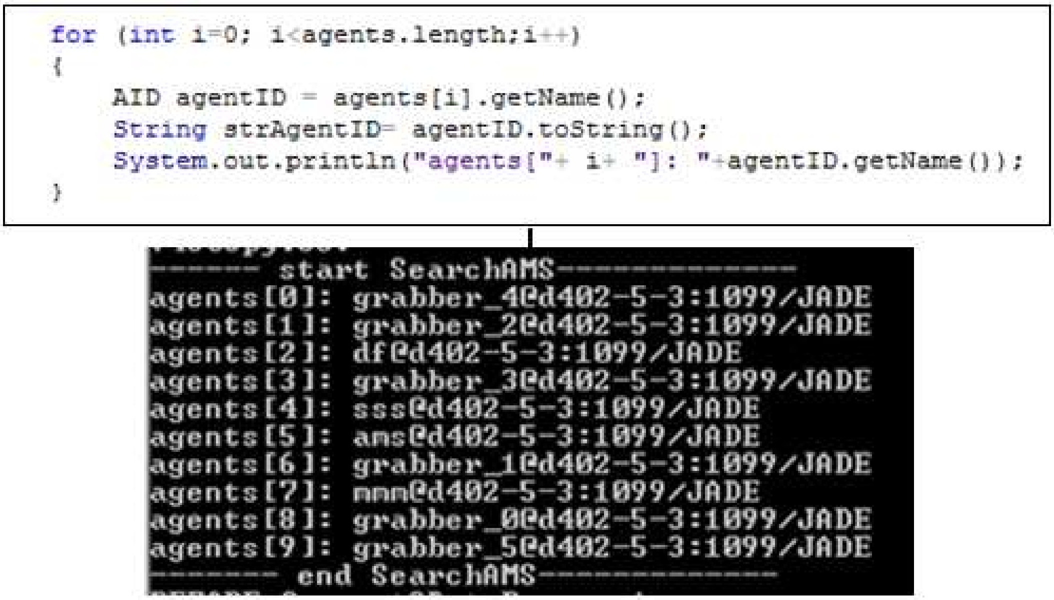


```
ACLMessage reply = Gmsg.createReply();
reply.setPerformative(ACLMessage.INFORM);
reply.setContent( "GrabberAgent: friend List has been sent in "+fileName);

//open FClient socket
openClientSocket();

send(reply);
```

Figure 6.12: Reply Message from *gAg* to *mAg*

In order to reply, the *gAg* uses the `createReply ( )` method and modifies the content and performative of the message. The `createReply ( )` method will automatically switch the sender and receiver and complete all necessary fields, such as the conversation-id (unique ID of the conversation thread), reply-with (the expression used by a responding agent, in this case a *gAg* , to identify the message) and in-reply-to (a reference to an earlier action to which the message is a reply), as shown in Figure 6.12.

When the *mAg* receives the reply message from its mail box accompanied with the file, it will save the information in a local repository. Part of the information



```
for (int i=0; i<agents.length;i++)
{
    AID agentID = agents[i].getName();
    String strAgentID= agentID.toString();
    System.out.println("agents["+ i+ "]: "+agentID.getName());
}
```

```
------ start SearchAMS-------------
agents[0]: grabber_4@d402-5-3:1099/JADE
agents[1]: grabber_2@d402-5-3:1099/JADE
agents[2]: df@d402-5-3:1099/JADE
agents[3]: grabber_3@d402-5-3:1099/JADE
agents[4]: sss@d402-5-3:1099/JADE
agents[5]: ams@d402-5-3:1099/JADE
agents[6]: grabber_1@d402-5-3:1099/JADE
agents[7]: mmm@d402-5-3:1099/JADE
agents[8]: grabber_0@d402-5-3:1099/JADE
agents[9]: grabber_5@d402-5-3:1099/JADE
------- end SearchAMS-------------
```

Figure 6.13: Results of Searching the Platform for *gAg*s

contains the list of friends. For each URL in the friends list, the *mAg* checks that it does not exist in the queue of the friends list before adding it at the rear. It then searches the platform for an available *gAg* . Figure 6.13 shows the code and a screen shot of a search of the platform for *gAg*s.

If a *gAg* is not found, the *mAg* will create a new *gAg* and then allocate it to the URL from the friends' list queue. Note that assigning a *gAg* to a URL forms a one to one relationship: i.e. each profile is visited and observed by only one *gAg* .

One of the strengths of a MAS is its agents' ability to work in parallel. Thus, the *mAg* uses parallel behaviour for allocating URLs to *gAg*s as shown in first line of the code in Figure 6.14. Thus, all *gAg*s will start to extract profiles information simultaneously.

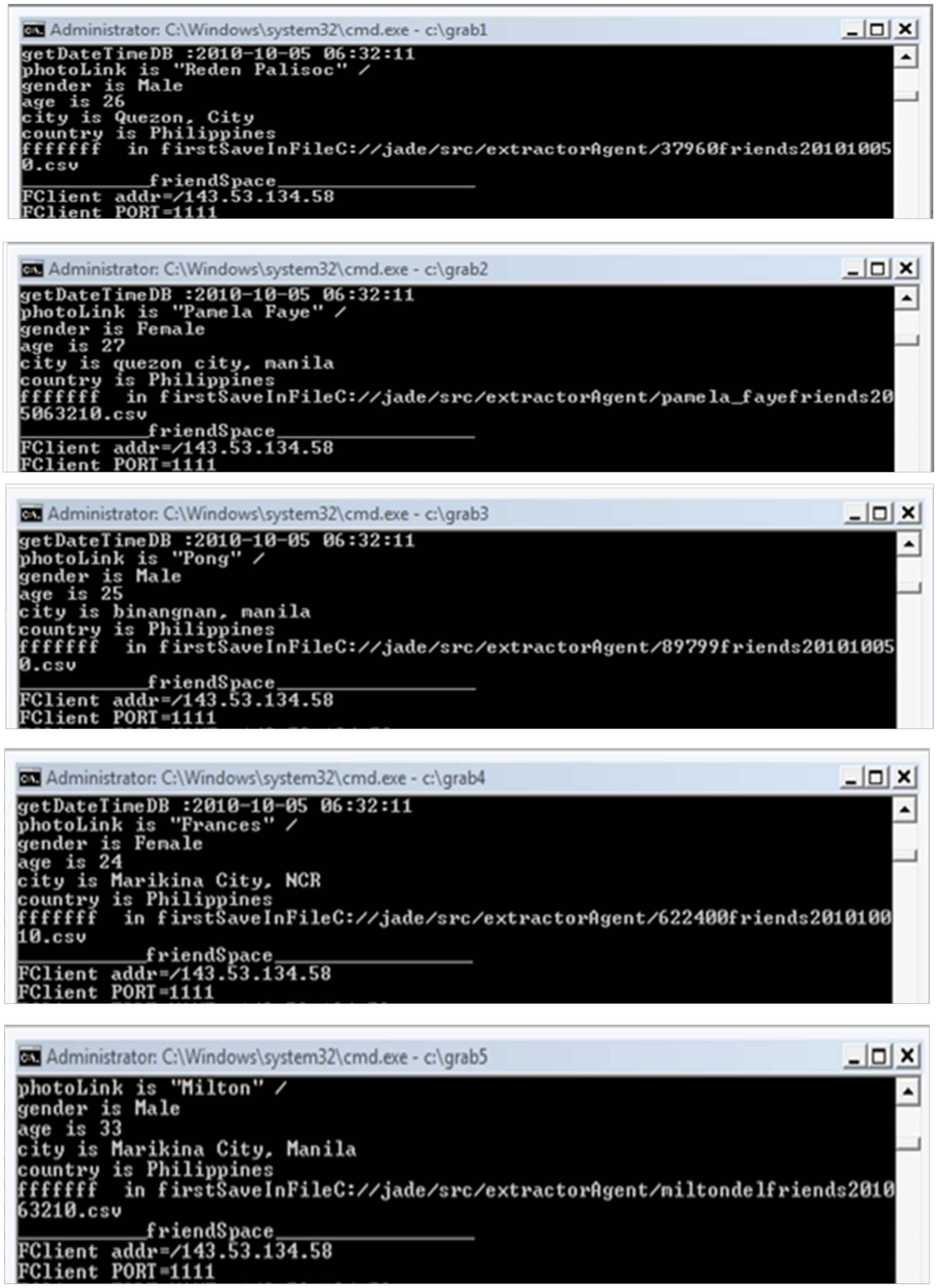
In order to keep an agent in each visited profile to track updates and send a message to the *mAg* containing any new changes, each *gAg* is created in its own container.

It should be noted that although all information is saved in a file to be sent to the *mAg* , only some of the basic information is printed out on the screen, along with the name of the file which contains the list of friends in Figure 6.15. In addition, although the list of top friends of the seed URL contains 7 friends, as shown in Figure 6.14, and the *mAg* sends a URL to each *gAg* , only 5 are valid, since the last two are excluded due to the fact that they are not related to an individual profile.

The process will continue using a BFS algorithm. Currently, the experiment is concerned with the top friends of each profile rather than all friends, for the reason specified in the previous Section 6.3.3. Through this process, the historical information from each profile will be saved in a local repository, keeping the *gAg* in the profile's URL to detect any changes in the profile contents.

The OSNRS terminates if the specified number of friends is retrieved or if all the

Figure 6.14: Parallel Behaviour of *mAg* in Allocating URL Profiles to *gAg*s



Administrator: C:\Windows\system32\cmd.exe - c:\grab1

```
getDateTimeDB :2010-10-05 06:32:11
photoLink is "Reden Palisoc" /
gender is Male
age is 26
city is Quezon, City
country is Philippines
fffffff  in firstSaveInFileC://jade/src/extractorAgent/37960friends20101005
0.csv
__________friendSpace________________
FClient addr=/143.53.134.58
FClient PORT=1111
```

Administrator: C:\Windows\system32\cmd.exe - c:\grab2

```
getDateTimeDB :2010-10-05 06:32:11
photoLink is "Pamela Faye" /
gender is Female
age is 27
city is quezon city, manila
country is Philippines
fffffff  in firstSaveInFileC://jade/src/extractorAgent/pamela_fayefriends20
5063210.csv
__________friendSpace________________
FClient addr=/143.53.134.58
FClient PORT=1111
```

Administrator: C:\Windows\system32\cmd.exe - c:\grab3

```
getDateTimeDB :2010-10-05 06:32:11
photoLink is "Pong" /
gender is Male
age is 25
city is binangnan, manila
country is Philippines
fffffff  in firstSaveInFileC://jade/src/extractorAgent/89799friends20101005
0.csv
__________friendSpace________________
FClient addr=/143.53.134.58
FClient PORT=1111
```

Administrator: C:\Windows\system32\cmd.exe - c:\grab4

```
getDateTimeDB :2010-10-05 06:32:11
photoLink is "Frances" /
gender is Female
age is 24
city is Marikina City, NCR
country is Philippines
fffffff  in firstSaveInFileC://jade/src/extractorAgent/622400friends2010100
10.csv
__________friendSpace________________
FClient addr=/143.53.134.58
FClient PORT=1111
```

Administrator: C:\Windows\system32\cmd.exe - c:\grab5

```
photoLink is "Milton" /
gender is Male
age is 33
city is Marikina City, Manila
country is Philippines
fffffff  in firstSaveInFileC://jade/src/extractorAgent/miltondelfriends2010
63210.csv
__________friendSpace________________
FClient addr=/143.53.134.58
FClient PORT=1111
```

Figure 6.15: Some of the Extracted Basic Information and the File of Friends List for Each *gAg* in its Own Container

friends at level 2 in the sub network have been retrieved.

6.3.5 Algorithm of OSNRS

Figure 6.16 outlines the pseudocode for the approach developed to extracting a profile's personal information and list of friends from the MySpace social network.

The described approach is applied in the two groups as described. The results of the experimental work are explained, followed by findings and future work.

Figure 6.16: The Pseudocode of the Developed OSNRS

6.4 Experimental Findings and Results

Since the target of this research focuses on justifying the approach developed rather than being concerned with the extracted data, a simple goal is set for the experiment in order to show the way in which the agent can track updates in OSN profiles. The goal set is to establish how often the users change their friends, either in the total number of friends or in their top list of close friends.

In Section 6.3.3, it was mentioned that two groups of Myspace profiles were selected. From the experiment, the number of public profiles observed in both groups ranged between 130 and 160 with changes in the sub network. Almost 80% of the visited profiles were aged below 30. Just over half of them (56%) were female.

Table 6.1 presents results obtained from preliminary analysis of the retrieved information. It shows that the user profile displays a significant difference between the minimum and maximum number of all friends. The mode of the number of top friends in both groups was similar, while in group 2 it was double regarding the number of all friends.

The profiles from both groups were observed for two weeks by reactivating the *gAg*s once a day. Tables 6.2 and 6.3 also show the daily changes in the number of friends list, top friends and all friends respectively. The empty cells at the end of

	Group 1				**Group 2**			
Total	84				59			
	Top Friends		**All Friends**		**Top Friends**		**All Friends**	
Min no. of Friends	1		3		3		9	
Max no. of Friends	39		535968		37		2150503	
Average of Friends	10.9		15190.64		12.55		54262.07	
Mode of Friends	6		20		7		43	
	Profiles	%	Profiles	%	Profiles	%	Profiles	%
Changed	1	1.19	20	24	1	2	19	42.22
Unchanged	60	71.43	37	44	35	78	17	37.78
Incomplete Changed	2	2.38	15	18	0	0	5	11.11
Incomplete Unhanged	18	21.43	11	13	9	20	4	8.89

Table 6.1: Comparison between Group 1 and Group 2

IDs	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14
1	6	6	6	6	6	6	6	6	6	6	6	6	6	6
2	7	7	7	7	7	7	7	7	7	7	7	7	7	7
3	16			16	16	16	16	16	16	16	16	16	16	16
4	25	25	25	25	25	25	25	25	25	25	25	25	25	
5	2	2	2	2	2	2	2	2	2	2	2	2	2	2
6	5	5	5	5	5	5	5	5	5	5	5	5	5	
7	8	8	8	8	8	8	8							
8	39	39	39	39	39	39	39	39	39	39	39	39	39	
9					5	5	5	5	5	5	5	5	5	5
10	11	11	11	11	11	11	11	11	11	11	11	11		
11	21	21	21	21	21	21	21	21	21	21	21	21	21	21
12	12	12	12	12	12	12	12	12	12	12	12	12	12	

Table 6.2: Sample of Tracking Number of Profiles Top Friends for 14 Days

IDs	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**	**Day 6**	**Day 7**	**Day 8**	**Day 9**	**Day 10**	**Day 11**	**Day 12**	**Day 13**	**Day 14**
1	9	9	9	11	12	12	12	12	12	12	12	12	12	12
2	34	34	34	34	34	34	34	34	34	34	34	34	34	34
3	154			154	154	154	154	154	154	154	154	154	154	154
4	312	312	312	312	312	312	312	312	312	312	312	312	312	
5	692	692	692	692	692	692	692	692	692	692	692	692	692	692
6	1165	1165	1165	1165	1165	1165	1165	1165	1165	1165	1165	1165	1165	
7	7595	7594	7593	7592	7591	7590	7589							
8	17487	17484	17476	17472	17465	17456	17451	17448	17581	17575	17568	17566	17565	
9					20808	20808	20806	20806	20806	20805	20803	20801	20800	20797
10	51888	51879	51871	51869	51862	51862	51852	51848	51837	51832	51828	51824		
11	65440	65427	65408	65399	65370	65347	65329	65315	65299	65276	65260	65249	65242	65230
12	459724	459610	459406	459321	459059	458788	461503	461402	461209	461611	461564	461430	461324	

Table 6.3: Sample of Tracking Number of All Friends for 14 Days

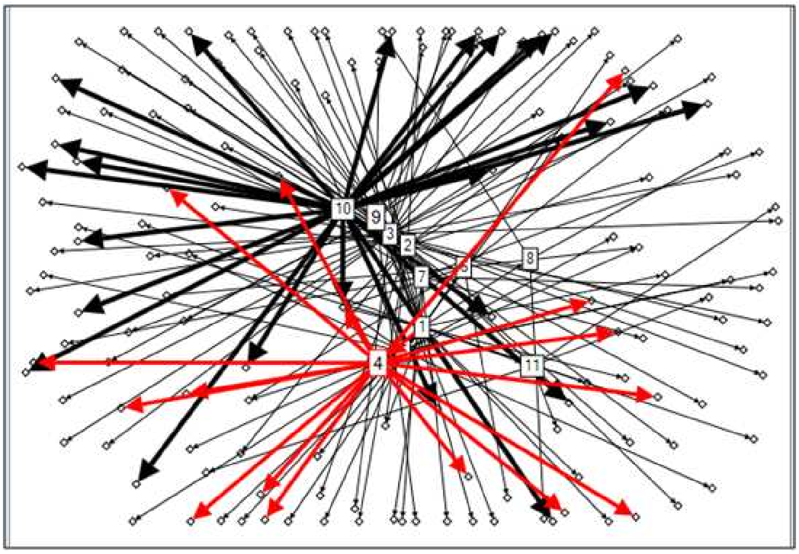


Figure 6.17: Sub Network of a Profile in Week 1

the row mean that either the profiles have been removed from the list of top friends of someone's profile, or that the account has been closed by the owner (e.g. profiles 4, 7, 10 and 12).

If the empty cells appear in the beginning of the table as in profile 9, this shows that this profile has been added recently to the list of top friends. The single most striking observation from the tables is profile number 3, which has been dropped from the list of top friends of one profile. Two days later, it appears again in the list of the top friends of another profile. The fact that the profile already exists in the database will undoubtedly help in understanding the impact of this profile on the sub network.

By combining Table 6.1 with Tables 6.2 and 6.3, it may be concluded that users are likely to change their friends list (adding or removing friends), but that they are unlikely to change their closer friends in their top list. In each group, less than 3% of users change their list of top friends.

As in Chapter 4, Nodexl is used to illustrate the friend networks of the profiles. Figure 6.17 and Figure 6.18 show examples of the changes in the sub network of a

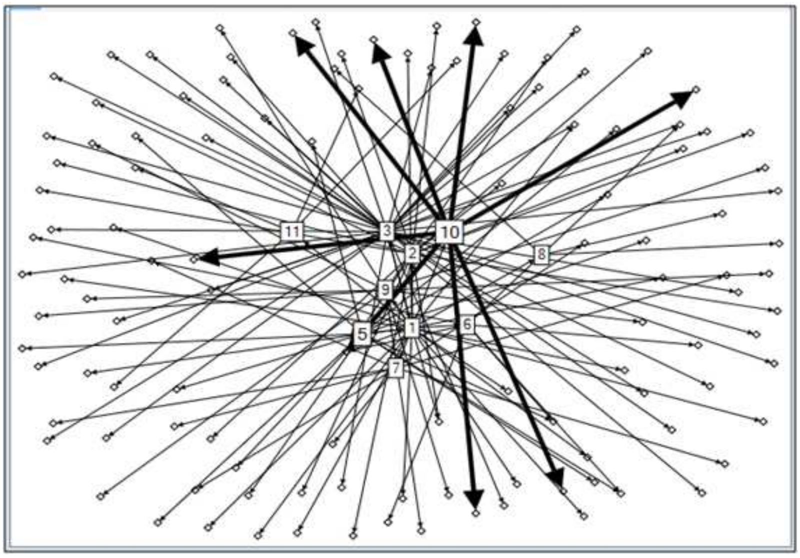


Figure 6.18: Sub Network of a Profile in Week 2

user profile in group 2 when its list of top friends is changed. From the two figures, it is noticed that the node labelled 4 in Figure 6.17 was removed from the friends' list in the second week of observation as shown in Figure 6.18. As a result, all friends of node 4 have been removed from the sub network. In addition, node number 10 has reduced its list of top friends in Figure 6.18. Such changes affect the resulting sub network and help in understanding user behaviour at the analysis stage.

Other observations may be made from Tables 6.4 and 6.5 which presents the number of public profiles in each top friends list, such as:

- A larger number of the friends list of a profile does not imply that a larger sub network can be retrieved. The key points depend on how many of those profiles are public. Moreover, missing information due to privacy issues will need to be factored into studies which analyze information from OSNs.

- Equality in the number of public profiles does not necessarily produce the same size of sub network. For example, in group 1, the top friends of profiles 5 and 6 are equals (15) with equal public profiles as well (3). However, the resulting sub networks significantly vary (38 and 3 respectively).

profile no.	**Top friends**	**Public profiles**	**Retrieved profiles**	**Public profiles in Top friends %**
1	5	4	56	80
2	7	7	74	100
3	2	1	8	50
4	4	1	1	25
5	15	3	38	20
6	15	3	3	20
7	4	4	22	100
8	35	11	151	31.43
9	10	5	40	50
10	5	3	15	60
11	6	5	78	83.33
12	6	4	17	66.67
13	3	2	13	66.67
14	14	10	79	71.43
15	6	5	46	83.33
16	7	6	48	85.71
17	6	6	30	100
18	12	10	135	83.33
19	3	2	20	66.67

Table 6.4: The First Iteration of Group 1 Percentage of Public Profile in Top Friends

The final but not the least finding in this experiment is that the result which is taken from a joint sample such as group 2 is more helpful in understanding the behaviour of the users of OSN than the results of disjointed users as in group 1. In fact, one of the means through which e-commerce companies target their customers is by advertising their products on OSN sites.

profile no.	**Top friends**	**Public profiles**	**Retrieved profiles**	**Public profiles in Top friends %**
1	11	10	50	90.91
2	11	7	109	63.64
3	30	20	100	66.67
4	2	0	0	0.00
5	12	6	40	50.00
6	7	7	122	100.00
7	30	15	155	50.00
8	6	2	21	33.33
9	6	4	58	66.67
10	6	3	23	50.00
11	11	2	56	18.18

Table 6.5: The First Iteration of Group 2 Percentage of Public Profile in Top Friends

6.5 Experimental Validation

This section shows that the application developed (OSNRS) has addressed the drawback of the previous approach which was presented in Chapter 4.

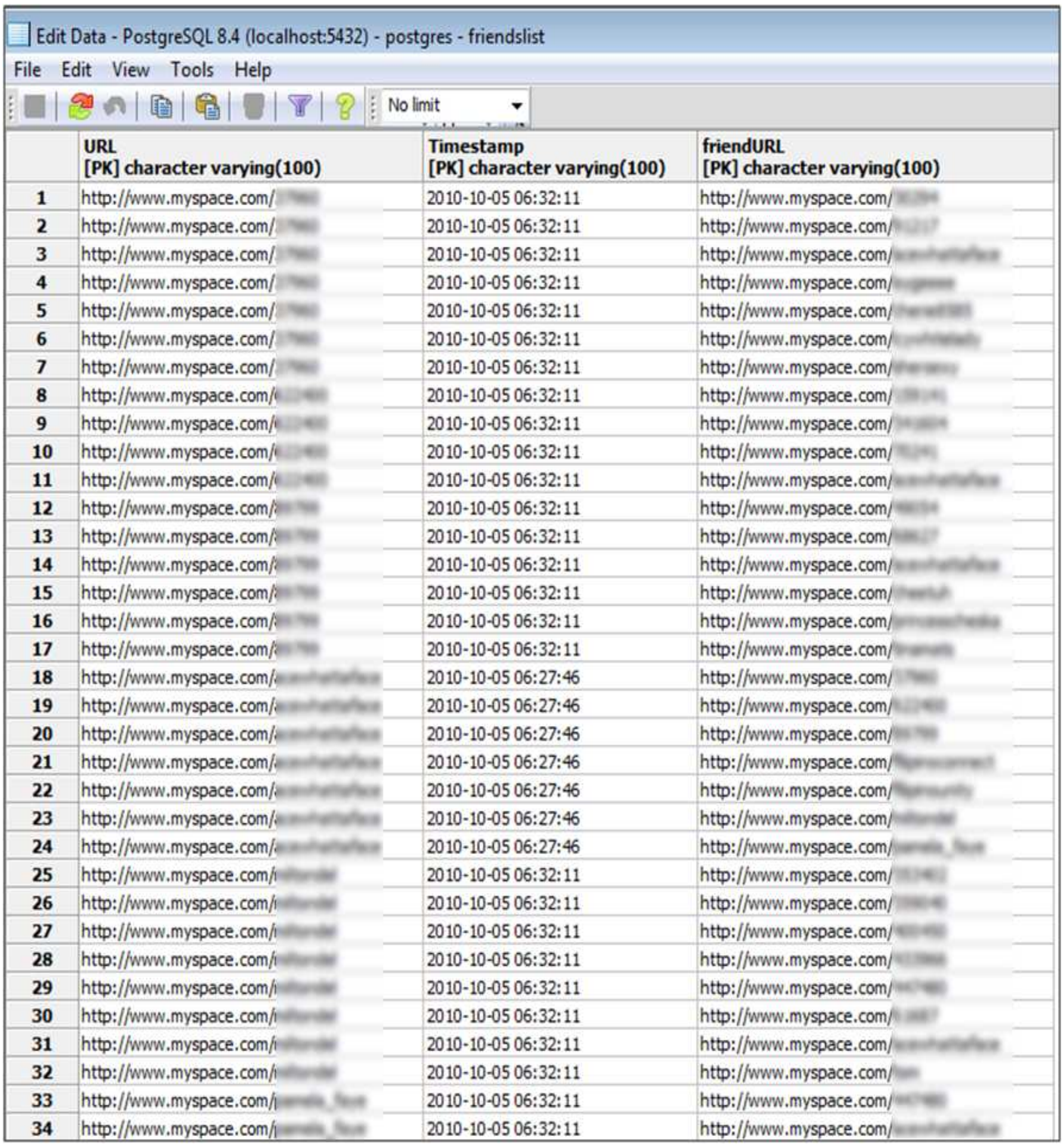
To evaluate the accuracy of the retrieved results, many profiles were tested manually by a co-author to be used in her research study of measuring vulnerability (see [8]. An example is the cases where unexpected results were reported, as shown in the case where empty cells in profile no. 3 in Tables 6.2 and 6.3. The results show that all the information retrieved is correct. However, future work to extend *mAg* functions to randomly check the contents of information retrieved at random times is ongoing.

The screen shots below shows a DB which connects each retrieved profile with its list of friends. Note that the time stamps for all retrieved files are equal because the *gAg*s are working in parallel, unlike the *gAg* which is allocated to the seed URL. Also if the time when the information is saved in the file in Figure 6.15 and the time when it is saved in the database in Figure 6.19 is compared, an almost 1 second delay can be seen for all files regarding to file transforming process.

6.6 Limitations

The process presented to correctly and accurately monitor and extract profile data is not affected by the size of the retrieved information in terms of DE techniques.

Also, further work is required in order to evaluate *mAg* functions in checking the contents of retrieved information to allow the *gAg* to decide the period of reactivation independently according to how often the profile's information is changed. Also, in order to save time and traffic capacity, the *gAg* should simply send the updated fields rather than the whole file.


Edit Data - PostgreSQL 8.4 (localhost:5432) - postgres - friendslist
File Edit View Tools Help
No limit

	URL [PK] character varying(100)	Timestamp [PK] character varying(100)	friendURL [PK] character varying(100)
1	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
2	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
3	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
4	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
5	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
6	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
7	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
8	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
9	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
10	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
11	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
12	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
13	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
14	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
15	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
16	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
17	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
18	http://www.myspace.com/[redacted]	2010-10-05 06:27:46	http://www.myspace.com/[redacted]
19	http://www.myspace.com/[redacted]	2010-10-05 06:27:46	http://www.myspace.com/[redacted]
20	http://www.myspace.com/[redacted]	2010-10-05 06:27:46	http://www.myspace.com/[redacted]
21	http://www.myspace.com/[redacted]	2010-10-05 06:27:46	http://www.myspace.com/[redacted]
22	http://www.myspace.com/[redacted]	2010-10-05 06:27:46	http://www.myspace.com/[redacted]
23	http://www.myspace.com/[redacted]	2010-10-05 06:27:46	http://www.myspace.com/[redacted]
24	http://www.myspace.com/[redacted]	2010-10-05 06:27:46	http://www.myspace.com/[redacted]
25	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
26	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
27	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
28	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
29	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
30	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
31	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
32	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
33	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]
34	http://www.myspace.com/[redacted]	2010-10-05 06:32:11	http://www.myspace.com/[redacted]

Figure 6.19: Snapshot of Friends List Table in Database

6.7 Chapter Summary

This chapter has investigated DE from OSNs through MAS technology implementation of OSNRS. OSNRS is designed with multiple agents each of which assigned to a specific profile. This work is an improvement on previous information extraction techniques from OSNs, in that the proposed algorithm allows the tracking of information changes in the users' profiles, while previous work was concerned with extracting information from profiles only once.

The experimental work shows that using MAS simplifies the process of tracking a profiles history. This study will serve as a base for future studies in the process of tracking user profile history and understanding the behaviour of OSN users, especially when combined with text mining.

The contribution to this chapter has been published in [13, 11, 8].

The next chapter will modify the implementation of OSNRS by using an Application Programming Interface (API), in order to come into alignment with recent changes to the structure of social network developing methods.

Chapter 7

Multi Agent System for Data Extraction from Online Social Network in Application Programming Interface Approach

7.1 Introduction

In the previous chapters, the developed approaches relied on the parser for the automated extraction of personal information of OSN profiles and their list of friends. However, the main drawback of the parser was its need for continuous modification to go along with any changes in the structure of the profiles' web page. Moreover, it would be useless if the profile's source documents were not provided by the OSN developers.

Most current web service providers, including OSN developers, offer what is called an Application Programming Interface (API) to allow software applications to communicate with each other accurately and securely over the web. Thus, this chapter aims to continue the previous work of OSNRS, which provides real time monitoring of OSN profiles, through proposing new algorithms using API in order to overcome the parser's drawback. This facility allows OSNRS agents to acquire the required attributes despite modifications in the representation of the profiles' source web pages.

The structure of remainder of the chapter is as follows: Section 7.2 introduces a conceptual overview for the use of API to improve OSNRS. Section 7.3 illustrates the Flow of information between OSNRS components with API while Section 7.4 describes the structural models of OSNRS with API. Section 7.5 presents different algorithms for integrating MAS with API. Section 7.6 explains the implementation of using API in extracting information from Facebook, currently the most popular OSN, supported with case studies. The findings and results are stated in Section 7.7. Finally, Section 7.8 summarizes the chapter.

7.2 Conceptual Overview of OSNRS with API

API is a software-to-software interface. Orenstein in [109] simplifies the definition of API to "*a description of the way one piece of software asks another program to perform a service. The service could be granting access to data or performing a specified function*".

In this case, OSNRS agents are going to connect with the OSNs providers using API to perform the task of extracting the attributes of interest from the OSN profiles rather than using a parser. The following subsections describe in detail the approach developed in using API programmatically to integrate it with OSNRS agents.

7.2.1 Specifying the OSNRS Domain

All previous contributions, which are presented in Chapters 4 and 6, have crawled Myspace web pages to extract profile information. Although the developers of Myspace provide great support to integrate its platform functionality with different applications including smart phones, Myspace will not be used as a domain for OSNRS application with API. Myspace has lost market share dramatically over the

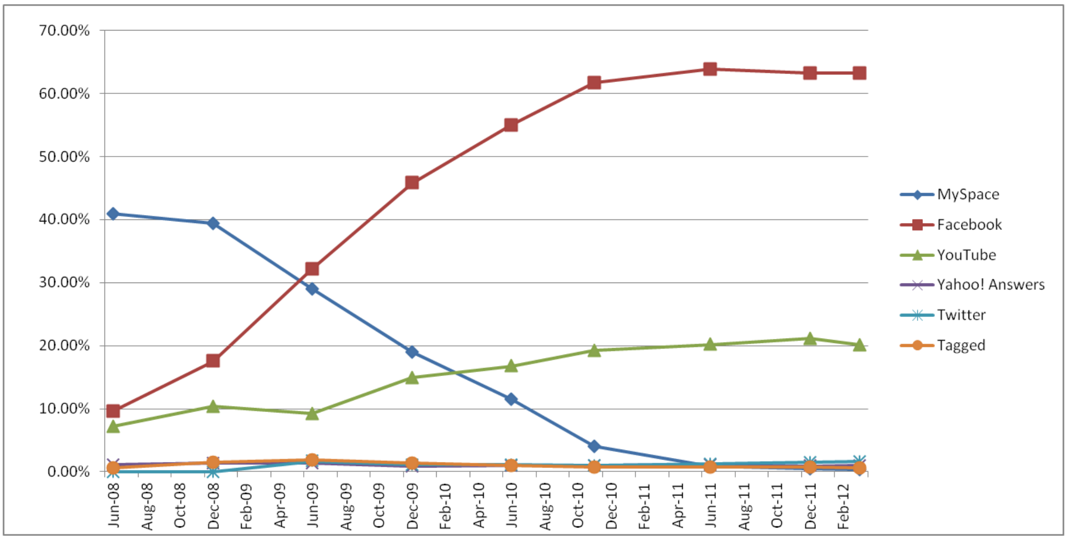


Figure 7.1: Top Online Social Network (Source of Data: Hitwise)

last three years. In contrast, Facebook has increased its share strongly as reported by [78] based on the United States market share of visits (see Figure 7.1). Facebook claimed by the end of January 2012 that their active users hit 8 million, as reported by BBC [29]. Therefore, Facebook was selected to be the domain for OSNRS data extraction using API.

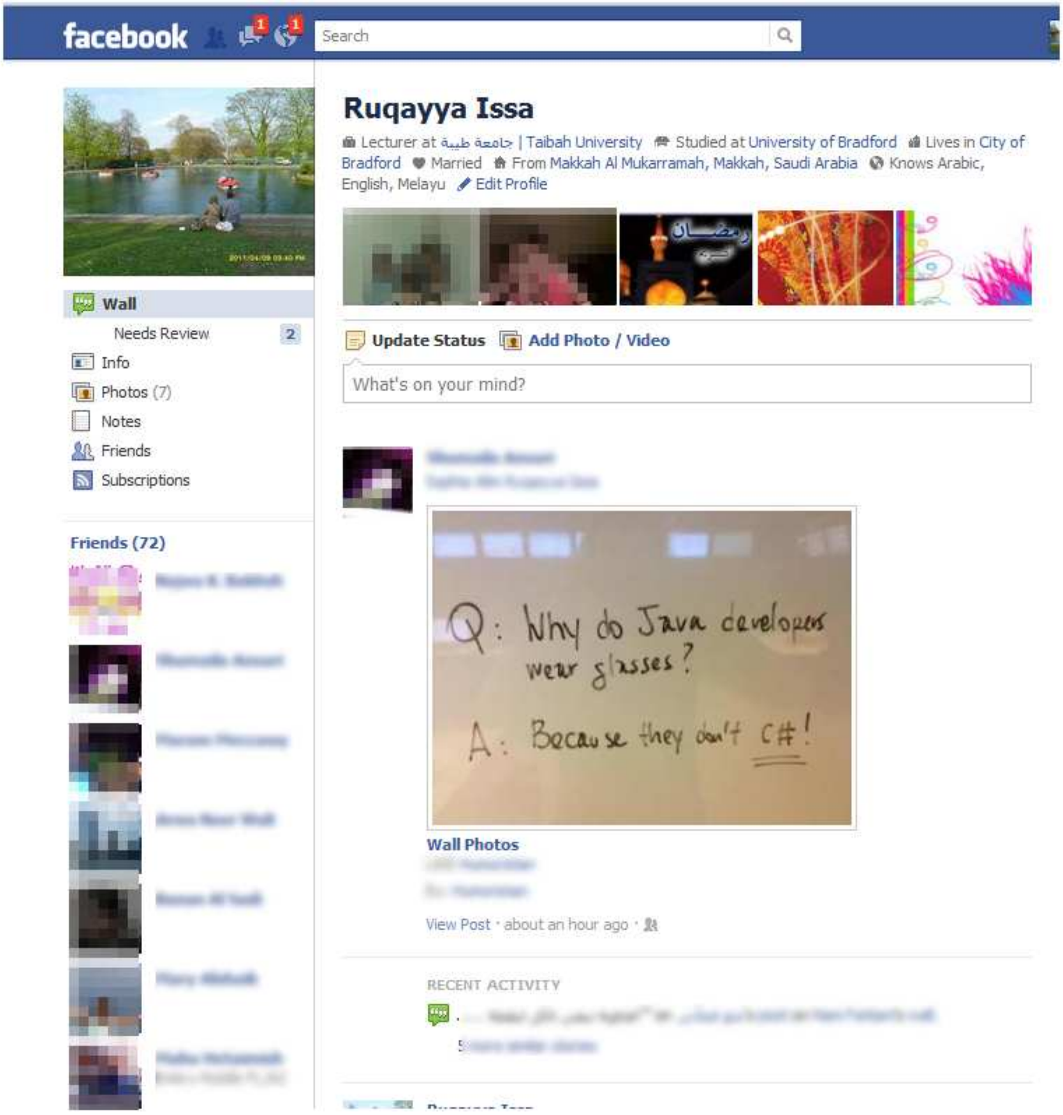
Facebook places users' privacy as the top priority. Thus, third-parties are not allowed to access users' profiles except through API. Facebook presents the Graph API as a core concept of its platform in order to allow applications access, to read and write data to it. Graph API considers users' profiles, photos, pages as objects, and friend relationships and photo tags as connections between objects. Note that attributes and connections differ between objects.(see Table 7.1)

Figure 7.2 shows a sample of a Facebook profile. Every object in Facebook has a unique ID. To retrieve all attributes of an object, request: `https://graph.facebook.com/ID` while `http://graph.facebook.com/ID/connection-type` is used to obtain the connection related to the object. E.g. the object "Page" called "Facebook platform" has an ID: 19292868552. Thus, to obtain the attributes of the object "Facebook platform" through Graph API, request `http://graph.`

User

Attributes(fields)		*Connections*	
public	*non − public*	*public*	*non − public*
id	likes	picture	accounts
name	third_party_id	permissions	achievements
first_name	timezone	payments	activities
middle_name	updated_time		albums
last_name	verified		apprequests
gender	bio		books
locale	birthday		checkins
link	education		events
username	email		family
	hometown		feed
	interested_in		friendlists
	location		friendrequests
	political		friends
	favourite_athletes		games
	favourite_team		groups
	quotes		home
	relationship_status		inbox
	religion		interests
	significant_other		likes
	video_upload_limits		links
	website		movies
	work		music
			mutualfriends
			notes
			notifications
			outbox
			photos
			pokes
			posts
			questions
			scores
			statuses
			subscribedto
			subscribers
			tagged
			television
			updates
			videos

Table 7.1: The Public and Non-Public Attributes and Connections of the Graph API Object User



facebook

Search

Ruqayya Issa

Lecturer at جامعة طيبة | Taibah University · Studied at University of Bradford · Lives in City of Bradford · Married · From Makkah Al Mukarramah, Makkah, Saudi Arabia · Knows Arabic, English, Melayu · Edit Profile

Wall

Needs Review 2

Info

Photos (7)

Notes

Friends

Subscriptions

Friends (72)

Update Status · Add Photo / Video

What's on your mind?

Q: Why do Java developers wear glasses?

A: Because they don't C#!

Wall Photos

View Post · about an hour ago ·

RECENT ACTIVITY

Figure 7.2: Sample of Facebook Profile

`facebook.com/19292868552` or `http://graph.facebook.com/platform` if the ID is the username of the object (a special case of the objects people and pages). To obtain the connection "links" of the Page object "Facebook platform", request `http://graph.facebook.com/19292868552/links&access_token`. The access token will be described in subsection 7.3. Graph API allows applications to read and write data to Facebook [58].

7.2.2 Creating a Facebook Application

In order to allow OSNRS agents to integrate with Facebook providers, a Facebook application called MY Software Agent (MYSA) is created as a desktop application rather than a web application in order to be compatible with the existing OSNRS through the following steps:

1. create an account in Facebook or log in if the account already exists.
2. join the developers' community.
3. build up the MYSA application.
4. authenticate MYSA to acquire two parameters: application ID (or API key) and application secret.
5. get the users' authorizations to allow MYSA to access their:
   - default information (e.g. public information).
   - more specific information (e.g. friends' basic information).

These steps will be described in detail in the coming sections.

7.3 The Flow of Information between OSNRS Components with API

Figure 7.3 describes the sequence of actions between OSNRS classes using API. The extraction process of OSNRS starts when the *mAg* , as the administrator of the MYSA application, obtains MYSA parameters (application ID and application

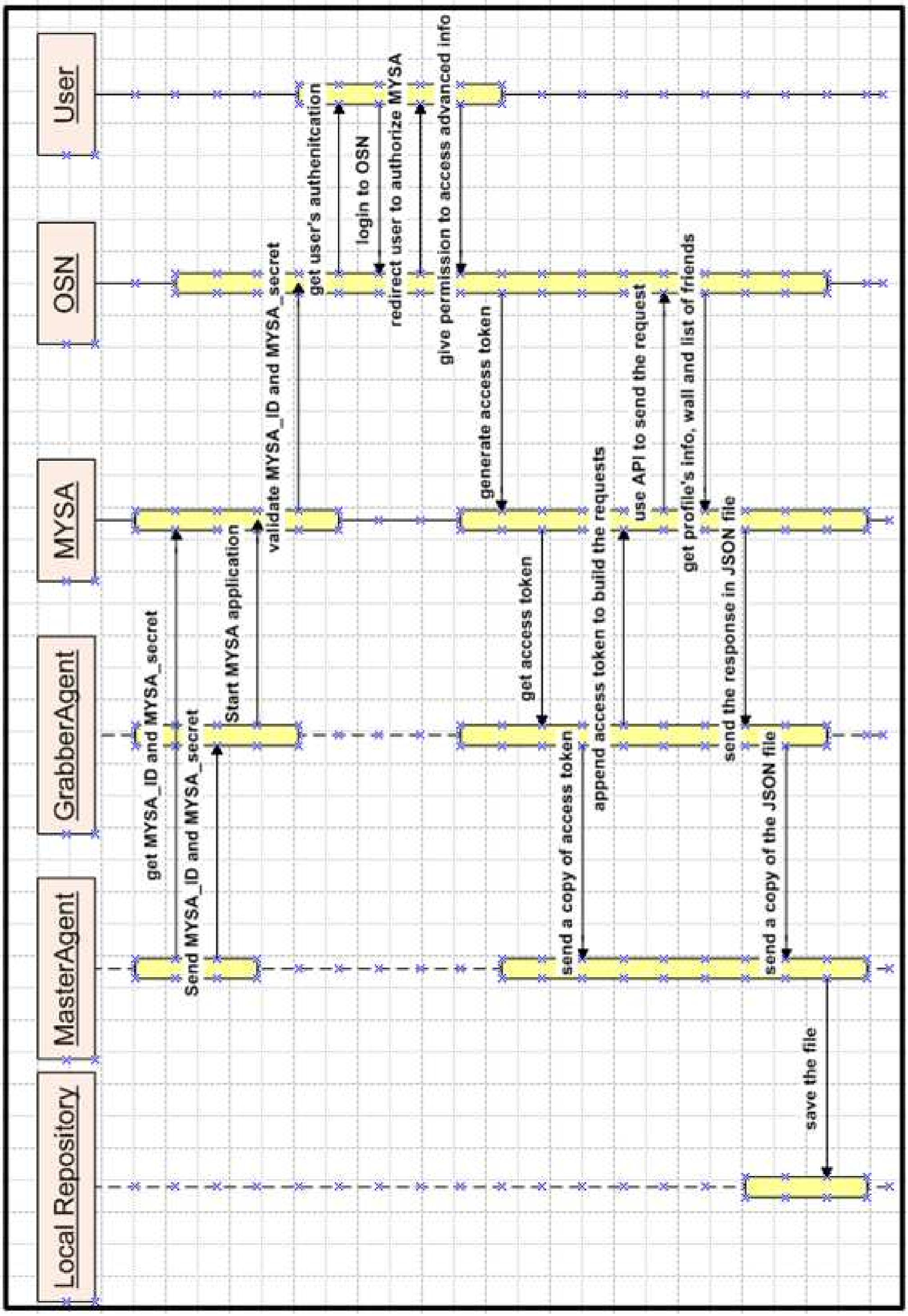
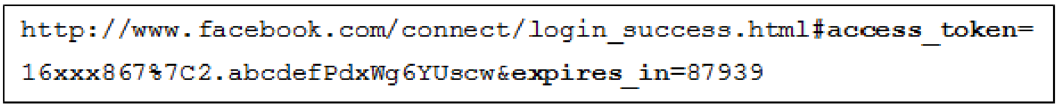

User
OSN
MYSA
GrabberAgent
MasterAgent
Local Repository
get MYSA_ID and MYSA_secret
Send MYSA_ID and MYSA_secret
Start MYSA application
validate MYSA_ID and MYSA_secret
get user's authenitcation
login to OSN
redirect user to authorize MYSA
give permission to access advanced info
generate access token
get access token
send a copy of access token
append access token to build the requests
use API to send the request
get profile's info, wall and list of friends
send the response in JSON file
send a copy of the JSON file
save the file

Figure 7.3: Sequence Diagram of OSNRS using API



```
http://www.facebook.com/connect/login_success.html#access_token=
16xxx867%7C2.abcdefPdxWg6YUscw&expires_in=87939
```

Figure 7.4: Sample of the Access Token

secret) in order to send them to a created *gAg* . Unlike previous approach in Chapter 6, the *gAg* has not yet been allocated to the seed profile.

Consequently, the *gAg* uses the received parameters to establish MYSA. MYSA's parameters should be validated by the OSN server (Facebook Server) to authenticate the user. The authentication process includes validation of the login cookies of the user, which are stored within the server if the user has already logged in. Otherwise, the user will be prompted to log in to OSN. The logged in user will be considered as the seed URL and will be allocated to the *gAg* .

Once the user has been authenticated successfully, the user will be redirected to authorize the application. Authorization means ensuring that the user knows exactly what type of data and permissions MYSA has been authorized to access. MYSA asks the user for extra permissions besides the basic information that Facebook provides by default, or which is specified as public information by the user.

Table 7.1 shows the public and non-public attributes and connections of the Facebook Graph API object "User". When the user authorizes MYSA, the application authentication step is approved and the OSN server will generate a response to the *gAg* . The response contains the access token accompanied with expiry parameter in seconds (See Figure 7.4).

The access token is the key for all requests using API. Thus the *gAg* will send a copy of it to the *mAg* , as well as using it to extract the information of the profile that it is allocated to. For example, the *gAg* will append the access token to build a request to extract the profile wall, and another request to obtain the friends list of the profile. The MYSA obtains the retrieved information from the Facebook server and sends it to the *gAg* in a file.

Facebook developers return the results either in common XML or JSON files. JSON format has been selected because it is smaller, faster and easier to parse than XML as shown previously in Section 2.3.3.

The file name combines the Facebook profile ID, the file contents such as info, wall, searchPeople, searchPost, etc and the time of retrieval in YYYYMMDDhhmmss format, similarly to the process explained in Section 6.3.4 (e.g. profileID-SearchPeople20120412082342). This is because each file name should be unique to distinguish which profile it belongs to and when it was retrieved in order to accomplish the aim of creating historical information for each profile.

As in the previous approaches, a copy of the file will be saved by the *gAg* for comparison, while another copy will be sent to the *mAg* to be saved in a local repository. Although database tables could be used to save the returned files as explained in Chapters 4 and 6, it was found to be inappropriate and time consuming to save the data retrieved as JSON files in database tables, then query the tables to extract data and save it back to a JSON file. Thus, JSON files are saved in the local repository.

For each friend in the friends list of the seed profile, different scenarios may be applied to monitor updates in the profiles, as shown in the next subsection.

7.4 Structural Models of OSNRS with API

Certain OSNs, including Facebook, do not allow their applications to access the friends of friends of the logged in user unless the user and his friends give their permissions. Thus, different models of OSNRS have been developed to explore how the *mAg* will control the process of extracting information from the seed profile's list of friends.

The models differ according to the proposed scenario. These scenarios will be compared to uncover the best algorithm in terms of the speed of the extraction process, the accuracy of the extracted information and the performance of the agents.

7.4.1 Scenario 1

This is the simplest scenario, where the *mAg* seeks the platform to find another *gAg* in order to extract and monitor all profiles in the friends list. I.e. in this case, one *gAg* is responsible for all profiles in the list of friends, as shown in Figure 7.5.

The *gAg* keeps a copy of information to use for comparison of results the next time the *gAg* is activated, and additionally the *gAg* can search on profiles' walls for keyword(s) that are required when mining data. For example, the agent could look for a keyword e.g. birthday, and then compare if the date of birth that is retrieved from the user's information matches the greetings date that friends wrote on the user's wall. The result would be used to help calculate the vulnerability of the user based on how the friends could leak the user's information, as explained in [8].

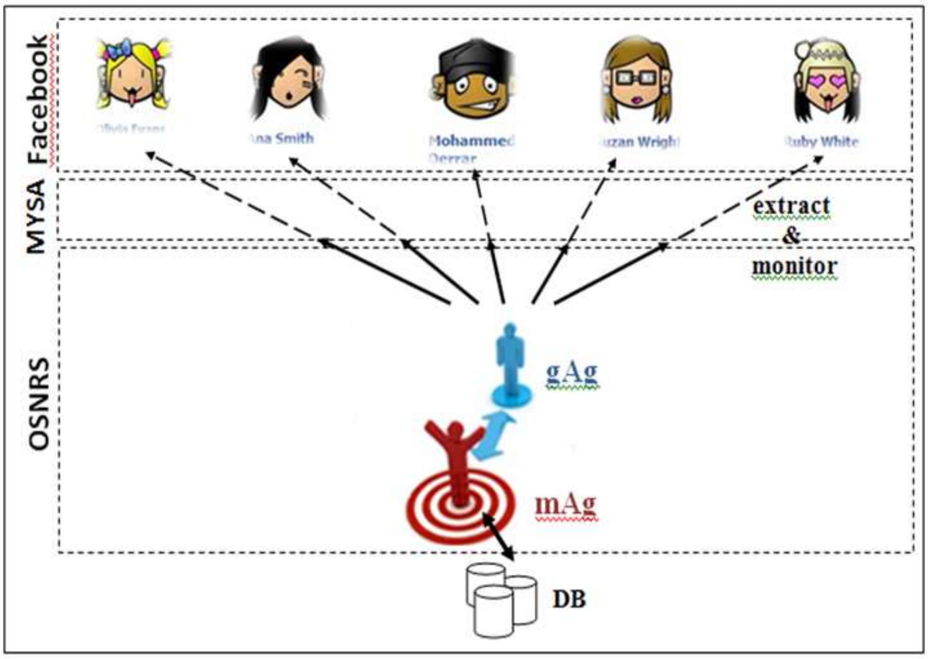


Figure 7.5: *gAg* for All Profiles

7.4.2 Scenario 2

The second scenario differs from the previous one in that when the *mAg* obtains the list of friends from the first *gAg* , it seeks the platform to assign a *gAg* to each profile in the friends list, as illustrated in Figure 7.6. If the existing *gAg*s are not sufficient, or are not found in first place, the *mAg* will create as many *gAg*s as required to match the number of friends in the list. Consequently, each *gAg* will act as the *gAg* in Section 7.4.1.

Returning to Section 7.3, which mentioned that the *mAg* acquires the access token from the *gAg* : besides the list of friends of the seed profile, the access token is required because the *gAg*s, which are allocated to each profile in the friends list, are not allowed to extract any information unless they have permissions from the seed profile. This permission can be gained through the access token.

For search purposes, OSNRS will create a special agent to look for the requested keyword(s). This agent will communicate with all *gAg*s in the platform. Having

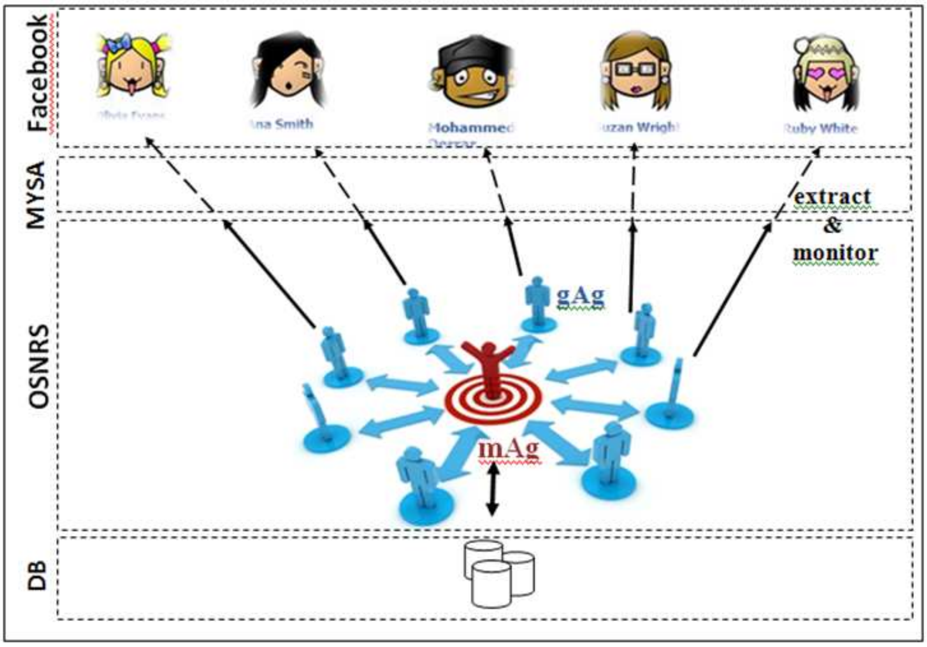


Figure 7.6: *gAg* for Each Profile

a special search agent is more appropriate than letting each *gAg* to scan all other walls to achieve the results.

7.4.3 Scenario 3

Since Facebook does not give permissions to applications to retrieve friends of friends lists, a scenario was proposed whereby two or more users could login to Facebook to allow MYSA to access their information. These users should be in the list of friends of the seed user. In this way, MYSA is allowed to move deeply into the social network indirectly.

For this scenario, there are two levels of control rather than one as there was in the previous scenarios. The *mAg* will control the whole system but instead of having control of all *gAg*s, a second level of control associated with the groupManagerAgent (*gmAg*) will be in the middle to coordinate communication between the *mAg* and the *gAg*s. See Figure 7.7.

Each agent, including *mAg* , *gAg* (s) and *gmAg* applies the autonomy feature

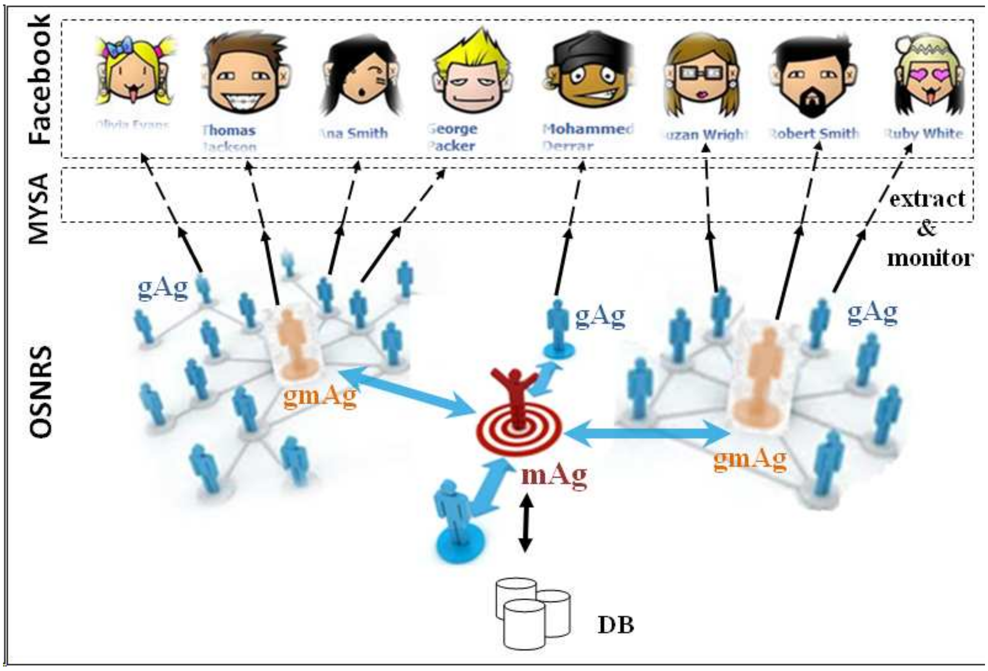


Figure 7.7: Two Levels of Controlling Profiles

through working as a standalone process in extracting data from a specific profile and monitoring its updates. Also, the autonomy feature of the *gAg* (including *gmAg*) allows it to decide on the suitable period of time to reactivate itself depending on how active the profile is.

The sociability feature is applied when agents communicate with each other (in addition to users) and exchange their knowledge in order to achieve their main shared goal, which is monitoring the OSN profiles over time. The perceptivity feature of the *gAg* is shown in that it can detect the real time updates of the profile that it is assigned to whenever it is activated.

7.5 Algorithm of OSNRS with API

The previous OSNRS algorithm which is explained in Section 6.3.5 relied on a parser to extract information from OSN web pages' source. Using such a parser is inconvenient or impossible in some cases where the OSN developers do not provide information except via an API.

In this section, algorithms are proposed to improve OSNRS by applying an API tool within the MAS approach. While Figure 7.8 describes the flow chart of OSNRS in the API approach, Figure 7.9 illustrates the pseudocode of the general algorithm of improved OSNRS that matches the sequence diagram explained in the previous subsection 7.3.

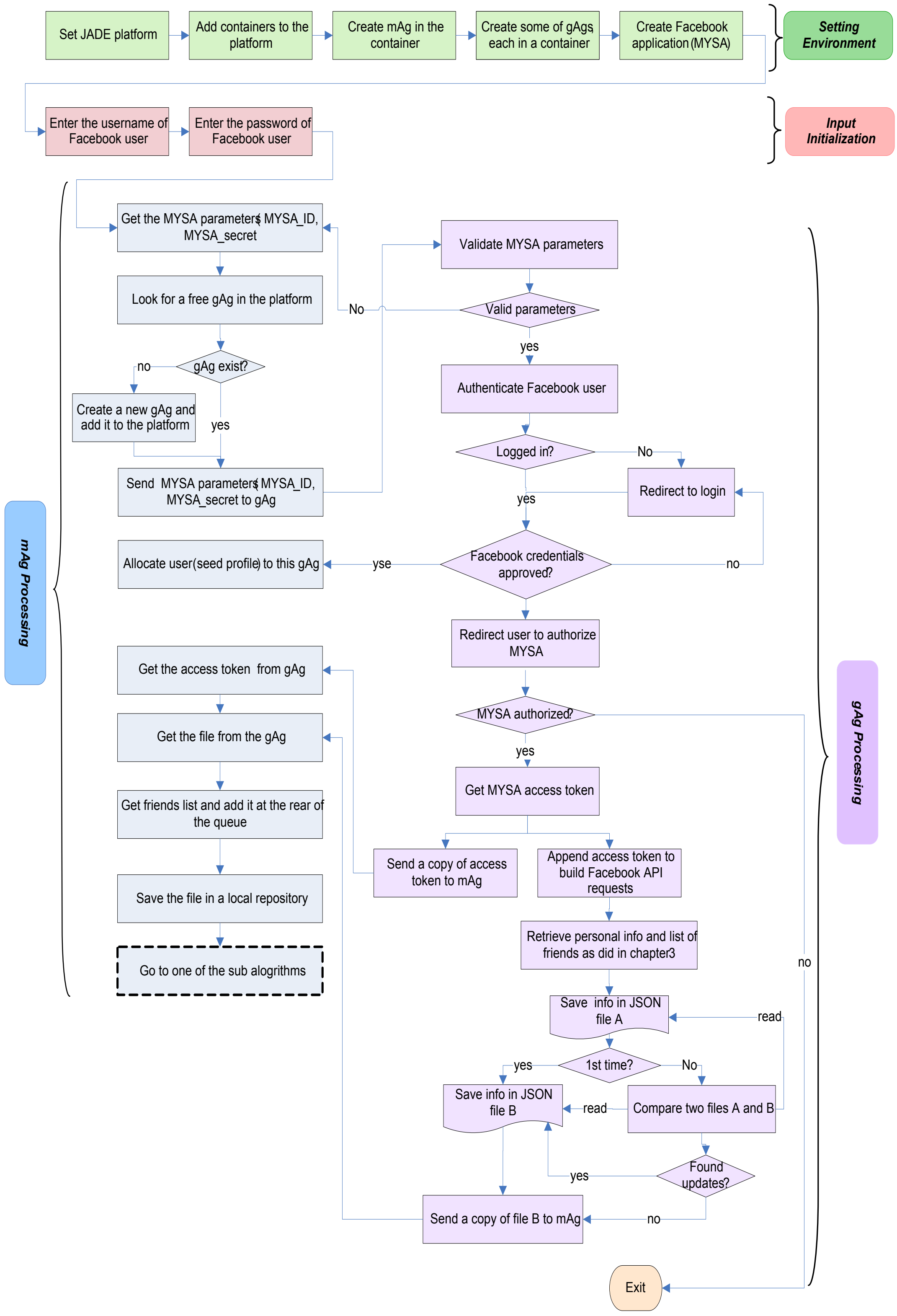
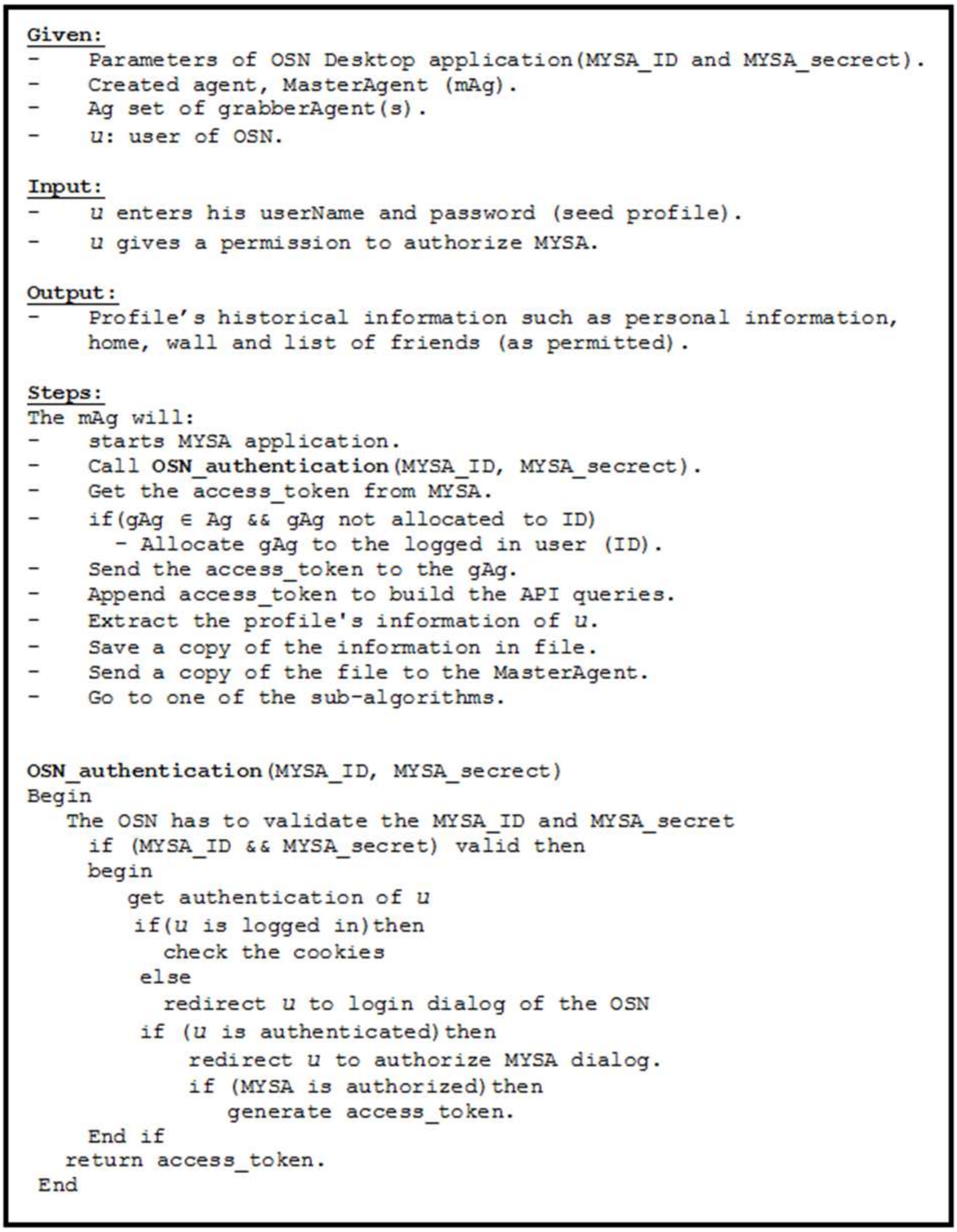

Set JADE platform
Add containers to the platform
Create mAg in the container
Create some of gAgs each in a container
Create Facebook application (MYSA)
Setting Environment
Enter the username of Facebook user
Enter the password of Facebook user
Input Initialization
mAg Processing
Get the MYSA parameters(MYSA_ID, MYSA_secret
Look for a free gAg in the platform
gAg exist?
no
Create a new gAg and add it to the platform
yes
Send MYSA parameters(MYSA_ID, MYSA_secret to gAg
Allocate user(seed profile) to this gAg
Get the access token from gAg
Get the file from the gAg
Get friends list and add it at the rear of the queue
Save the file in a local repository
Go to one of the sub alogrithms
Validate MYSA parameters
Valid parameters
No
yes
Authenticate Facebook user
Logged in?
No
Redirect to login
yes
Facebook credentials approved?
yse
no
Redirect user to authorize MYSA
MYSA authorized?
yes
Get MYSA access token
Send a copy of access token to mAg
Append access token to build Facebook API requests
Retrieve personal info and list of friends as did in chapter3
Save info in JSON file A
read
1st time?
yes
No
Save info in JSON file B
read
Compare two files A and B
Found updates?
yes
no
Send a copy of file B to mAg
no
Exit
gAg Processing

Figure 7.8: Flowchart of General Algorithm of OSNRS with API

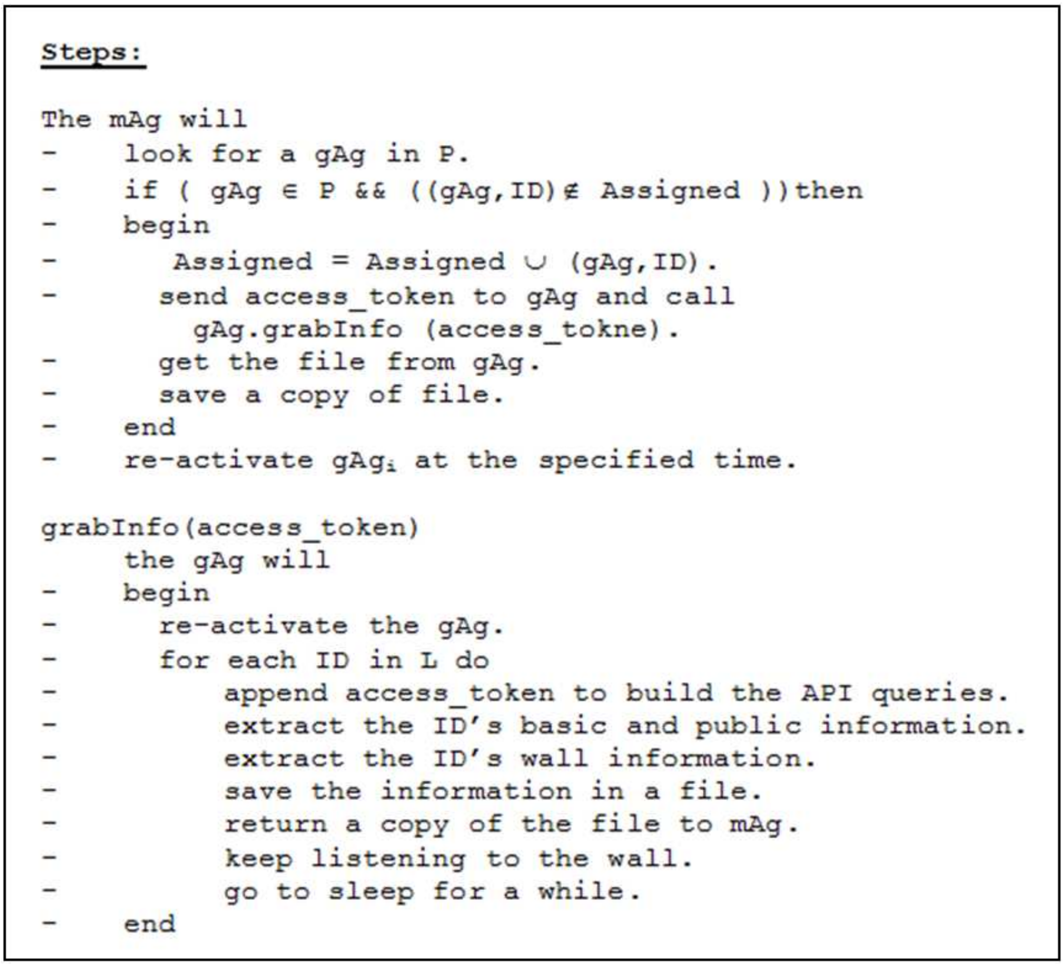


```
Given:
-    Parameters of OSN Desktop application(MYSA_ID and MYSA_secrect).
-    Created agent, MasterAgent (mAg).
-    Ag set of grabberAgent(s).
-    u: user of OSN.

Input:
-    u enters his userName and password (seed profile).
-    u gives a permission to authorize MYSA.

Output:
-    Profile's historical information such as personal information,
     home, wall and list of friends (as permitted).

Steps:
The mAg will:
-    starts MYSA application.
-    Call OSN_authentication(MYSA_ID, MYSA_secrect).
-    Get the access_token from MYSA.
-    if(gAg ∈ Ag && gAg not allocated to ID)
       - Allocate gAg to the logged in user (ID).
-    Send the access_token to the gAg.
-    Append access_token to build the API queries.
-    Extract the profile's information of u.
-    Save a copy of the information in file.
-    Send a copy of the file to the MasterAgent.
-    Go to one of the sub-algorithms.


OSN_authentication(MYSA_ID, MYSA_secrect)
Begin
   The OSN has to validate the MYSA_ID and MYSA_secret
     if (MYSA_ID && MYSA_secret) valid then
     begin
        get authentication of u
        if(u is logged in)then
          check the cookies
         else
          redirect u to login dialog of the OSN
         if (u is authenticated)then
            redirect u to authorize MYSA dialog.
            if (MYSA is authorized)then
               generate access_token.
     End if
   return access_token.
 End
```

Figure 7.9: General Algorithm of OSNRS with API

7.5.1 Sub Algorithm 1

In the first sub algorithm, as illustrated in Figure 7.10, when the *mAg* obtains the list of the profile's friends, it will assign one *gAg* to retrieve all possible information and to monitor the updates on the profiles of all friends in the user's list of friends. These updates will be sent back to the *mAg* to build the history of each profile.

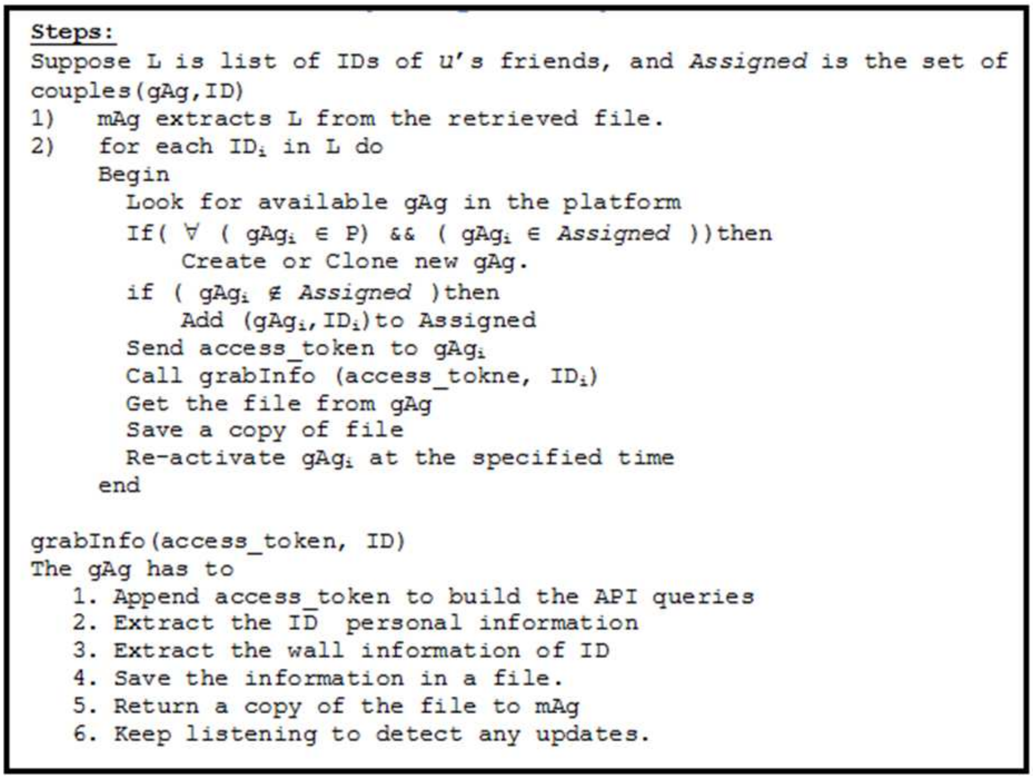


```
Steps:

The mAg will
-     look for a gAg in P.
-     if ( gAg ∈ P && ((gAg,ID)∉ Assigned ))then
-     begin
-        Assigned = Assigned ∪ (gAg,ID).
-       send access_token to gAg and call
          gAg.grabInfo (access_tokne).
-       get the file from gAg.
-       save a copy of file.
-     end
-     re-activate gAg$_i$ at the specified time.

grabInfo(access_token)
      the gAg will
-     begin
-       re-activate the gAg.
-       for each ID in L do
-           append access_token to build the API queries.
-           extract the ID's basic and public information.
-           extract the ID's wall information.
-           save the information in a file.
-           return a copy of the file to mAg.
-           keep listening to the wall.
-           go to sleep for a while.
-     end
```

Figure 7.10: OSNRS Sub Algorithm 1

7.5.2 Sub Algorithm 2

In the second sub algorithm, as shown in Figure 7.11, the *mAg* will assign a *gAg* to each friend in the user's list of friends. These *gAg*s will communicate with each other to exchange knowledge as will be explained in the experimental work. Again, all information will be sent back to the *mAg* to build the history.



```
Steps:
Suppose L is list of IDs of U's friends, and Assigned is the set of
couples(gAg,ID)
1)   mAg extracts L from the retrieved file.
2)   for each $ID_i$ in L do
     Begin
       Look for available gAg in the platform
       If( $\forall$ ( $gAg_i \in P$) && ( $gAg_i \in Assigned$ ))then
           Create or Clone new gAg.
       if ( $gAg_i \notin Assigned$ )then
           Add ($gAg_i$,$ID_i$)to Assigned
       Send access_token to $gAg_i$
       Call grabInfo (access_tokne, $ID_i$)
       Get the file from gAg
       Save a copy of file
       Re-activate $gAg_i$ at the specified time
     end

grabInfo(access_token, ID)
The gAg has to
   1. Append access_token to build the API queries
   2. Extract the ID  personal information
   3. Extract the wall information of ID
   4. Save the information in a file.
   5. Return a copy of the file to mAg
   6. Keep listening to detect any updates.
```

Figure 7.11: OSNRS Sub Algorithm 2

7.5.3 Sub Algorithm 3

This algorithm is a combination of the two previous algorithms. It has two levels of control. When the *mAg* receives the list of friends it will assign it to a special kind of *gAg* called a groupManagerAgent (*gmAg*). The *gmAg* has to play two roles. Firstly, it will extract the information of the profile of the friend in the seed's friends list. Secondly, it will start to work as a *mAg* for the friends list of this profile. Thus, the *mAg* will control the *gmAg* directly and all other *gAg*s indirectly. Also, the *gmAg* must send the list of friends to the *mAg* to obtain its permission before assigning any *gAg* to the profiles.

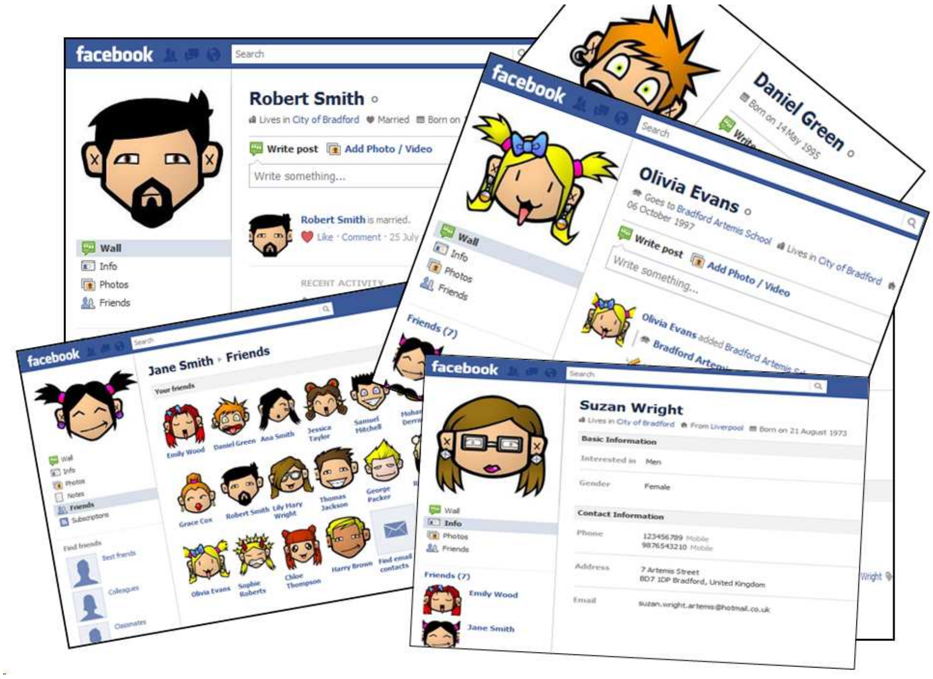
Note that the first two algorithms have been implemented in this thesis while the third algorithm is stated as a future work.

7.6 Implementation of OSNRS with API

The experimental work presents different case studies based on various scenarios, which are explained earlier, to run the OSNRS. However, some common steps should be set up for these case studies as follows:

7.6.1 Setting up the OSNRS Environment

The OSNRS environment has set up as was done in Section 6.3.1 using JADE. The additional setting in this experiment is related to using API. Facebook has the users' privacy as their top priority. Thus, third-parties are not allowed to access users' profiles aside from through API. To use Facebook Graph API, a Facebook (either desktop or web) application must be created. For the thesis, a Facebook application named MYSA is created as a desktop application rather than a web application, to be compatible with the OSNRS developed in the previous chapter. Facebook demands its applications, e.g. MYSA, to be authorized and authenticated



facebook Search

Robert Smith ○

Lives in City of Bradford · Married · Born on

Write post · Add Photo / Video

Write something...

Robert Smith is married.

Like · Comment · 25 July

RECENT ACTIVITY

Wall · Info · Photos · Friends

Daniel Green ○

Born on 14 May 1995

Olivia Evans ○

Goes to Bradford Artemis School · Lives in City of Bradford · 06 October 1997

Write post · Add Photo / Video

Write something...

Olivia Evans added Bradford Artemis School

Wall · Info · Photos · Friends

Friends (7)

Jane Smith › Friends

Your friends

Emily Wood, Daniel Green, Ana Smith, Jessica Taylor, Samuel Mitchell, Mohan Derra..., Grace Cox, Robert Smith, Lily Mary Wright, Thomas Jackson, George Packer, Olivia Evans, Sophie Roberts, Chloe Thompson, Harry Brown, Find email contacts

Wall · Info · Photos · Notes · Friends · Subscriptions

Find friends: Best friends · Colleagues · Classmates

facebook Search

Suzan Wright

Lives in City of Bradford · From Liverpool · Born on 21 August 1973

Basic Information

Interested in: Men

Gender: Female

Contact Information

Phone: 123456789 Mobile
9876543210 Mobile

Address: 7 Artemis Street
BD7 1DP Bradford, United Kingdom

Email: suzan.wright.artemis@hotmail.co.uk

Wall · Info · Photos · Friends

Friends (7)

Emily Wood

Jane Smith

Figure 7.12: A Sample of Mock Network Profiles

by their users to ensure that the users give their permission to MYSA to access their information.

7.6.2 Choosing the OSNRS Sample

The experiment has been tested on the real Facebook accounts of the author and her friends networks, as well as a mock network consisting of more than 20 Facebook users which has been used as an OSNRS sample.

The mock network is taken from a project called “The Artemis” developed at the University of Bradford. Figure 7.12 shows a screenshot of some of the mock network profiles. The users of the mock network are connected partially with each other in a random relationship.

Figure 7.13 shows the mock network where the vertices represent the users or profiles and the edges are the relationships. Note that just over half of the sample (55%) are females, symbolized by rectangle vertices, while males are symbolized by circle vertices.

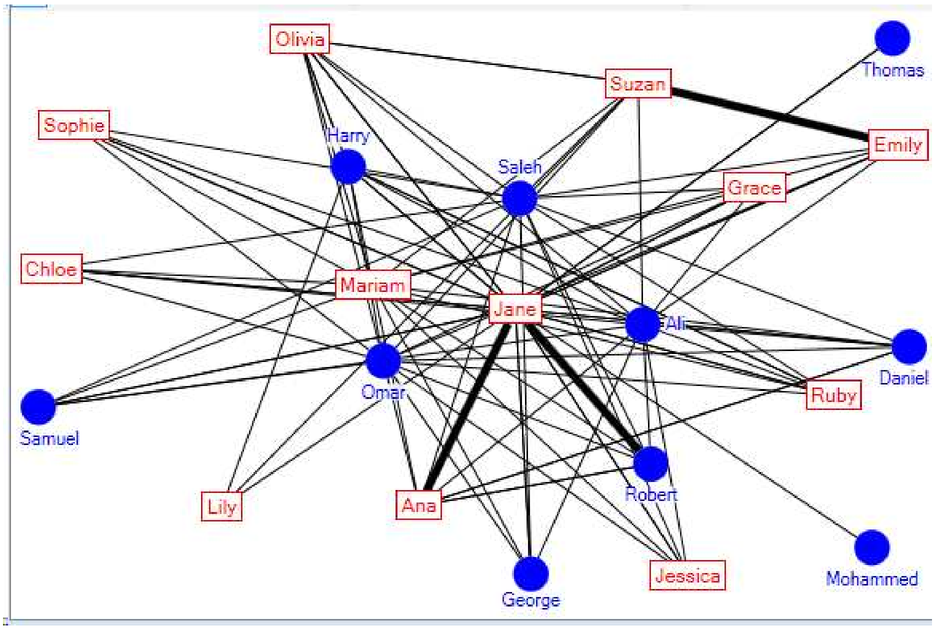
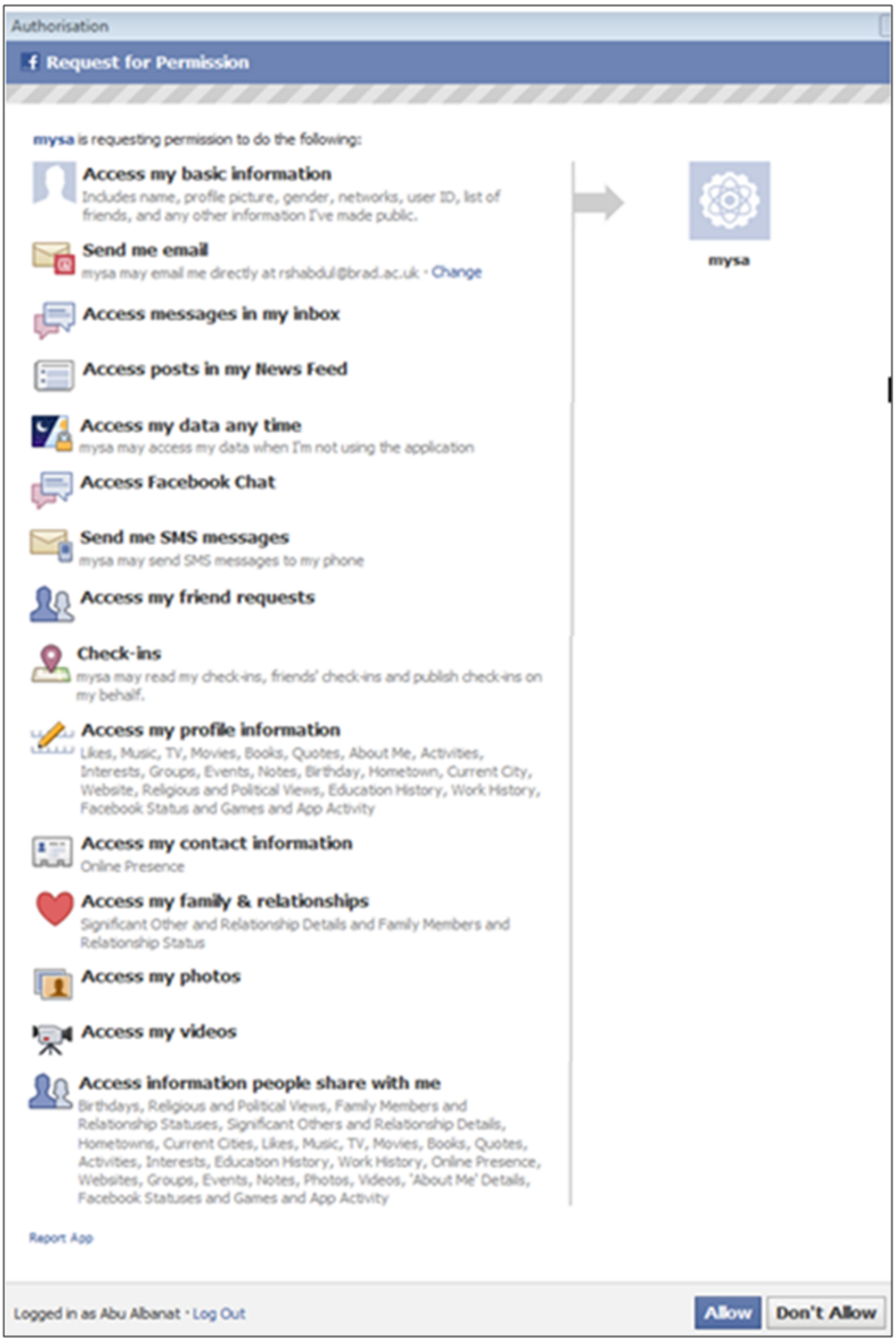


Figure 7.13: Mock Network for OSNRS

41% of users are stated as classmates while 22% are connected by a family relationship such as parent and child, or sibling (represented by bold edges). The rest are friends or have shared interests.

7.6.3 Running the OSNRS with API

Following the setting up of the JADE environment and the building of the MYSA application, the actual running of OSNRS is established when the *mAg* is created and looks for an existing *gAg* to send the MYSA's ID and secret key. The process continues as described previously in Sections 7.3 and 7.4. Notice that besides the basic information which Facebook developers provide by default, and the information that Facebook users set as public, MYSA asks users for permission to access extra information, as illustrated in Figure 7.14 . From the point where the *gAg* acquires the access token through Facebook Graph API, different case studies are given to manage how the retrieved information of the first user (seed profile) may be handled.



Authorisation

Request for Permission

mysa is requesting permission to do the following:

Access my basic information
Includes name, profile picture, gender, networks, user ID, list of friends, and any other information I've made public.

Send me email
mysa may email me directly at rshabdul@brad.ac.uk · Change

Access messages in my inbox

Access posts in my News Feed

Access my data any time
mysa may access my data when I'm not using the application

Access Facebook Chat

Send me SMS messages
mysa may send SMS messages to my phone

Access my friend requests

Check-ins
mysa may read my check-ins, friends' check-ins and publish check-ins on my behalf.

Access my profile information
Likes, Music, TV, Movies, Books, Quotes, About Me, Activities, Interests, Groups, Events, Notes, Birthday, Hometown, Current City, Website, Religious and Political Views, Education History, Work History, Facebook Status and Games and App Activity

Access my contact information
Online Presence

Access my family & relationships
Significant Other and Relationship Details and Family Members and Relationship Status

Access my photos

Access my videos

Access information people share with me
Birthdays, Religious and Political Views, Family Members and Relationship Statuses, Significant Others and Relationship Details, Hometowns, Current Cities, Likes, Music, TV, Movies, Books, Quotes, Activities, Interests, Education History, Work History, Online Presence, Websites, Groups, Events, Notes, Photos, Videos, 'About Me' Details, Facebook Statuses and Games and App Activity

Report App

Logged in as Abu Albanat · Log Out

Figure 7.14: MYSA's Authorization Request

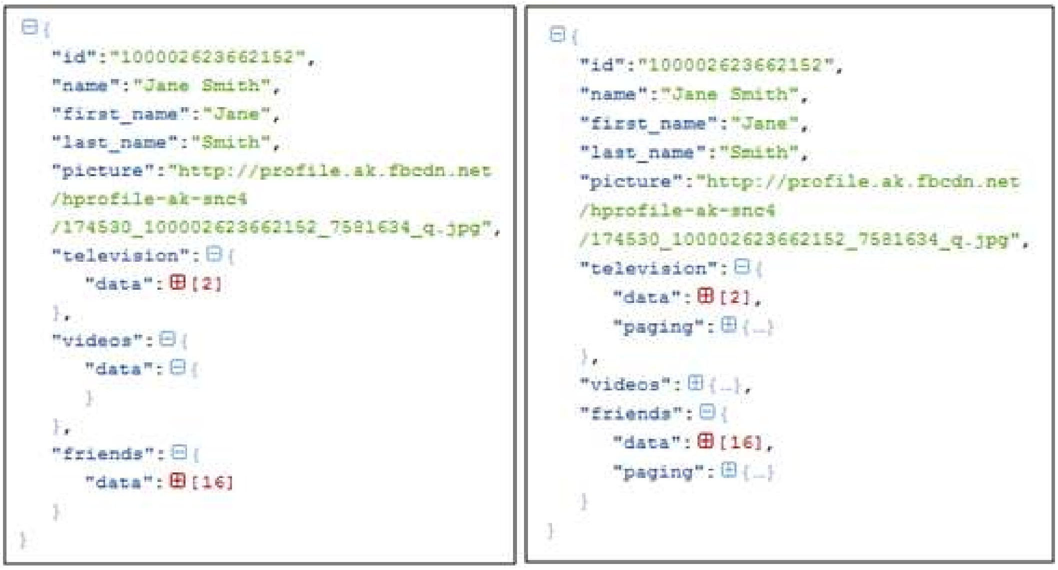
Criterion	Parser	API
Allow accessing Friends of Friends (FoF) automatically	✓	X
Allow direct information access	X	✓
More concern about privacy	X	✓
Require updating system regarding structure's changes	✓	X
Allow accessing basic information (even for private profiles)	✓	✓

Table 7.2: Table 2 Parser vs API in Developing OSNRS

7.7 Findings and Results

The aim of the experimental work is to improve OSNRS using the API tool. Thus, Table 7.2 shows some of the differences between retrieving information using the parser in the first version of OSNRS, which is implemented in Chapter 6, and in the improved version of OSNRS in this paper which retrieves information using API.

Each of parser and API has some advantages and dis advantages in such away that while the parser allows accessing FOF automatically, API concerns more about privacy and allows accessing information directly without being affected by the changes in the profile's structure. I.e. DE using API is faster, more precise and accurate in terms of data extracted.



```
⊟{
   "id":"100002623662152",
   "name":"Jane Smith",
   "first_name":"Jane",
   "last_name":"Smith",
   "picture":"http://profile.ak.fbcdn.net
   /hprofile-ak-snc4
   /174530_100002623662152_7581634_q.jpg",
   "television":⊟{
      "data":⊞[2]
   },
   "videos":⊟{
      "data":⊟{
      }
   },
   "friends":⊟{
      "data":⊞[16]
   }
}
```

```
⊟{
   "id":"100002623662152",
   "name":"Jane Smith",
   "first_name":"Jane",
   "last_name":"Smith",
   "picture":"http://profile.ak.fbcdn.net
   /hprofile-ak-snc4
   /174530_100002623662152_7581634_q.jpg",
   "television":⊟{
      "data":⊞[2],
      "paging":⊞{…}
   },
   "videos":⊞{…},
   "friends":⊟{
      "data":⊞[16],
      "paging":⊞{…}
   }
}
```

Figure 7.15: Parts of JSON Results (Week 1 on the Left, Week 2 on the Right)

Seed Profile	No of Friends	Total Files (Unit)	Total Size(KB)	**Total Process (Seconds)**	
				Case Study 1	Case Study 2
1	7	16	61	66	45
2	5	12	56	51	42
3	6	14	60	58	42
4	5	12	56	52	43
5	5	12	50	51	39
6	5	12	56	53	43
7	5	12	56	53	90
8	8	18	75	82	50
9	6	14	58	62	44
10	6	14	58	50	49
11	7	17	60	71	45
12	7	16	62	64	49
13	5	12	56	53	40
14	10	34	74	119	79
15	6	14	60	59	41

Table 7.3: Some Results of Case Studies 1 and 2 on Mock Network

Although generally speaking, there is no need to be concerned about the structure of data representation when API is used, it was surprising that OSNRS code still needed to be updated to deal with the changes in the structure of data in the JSON file.

Figure 7.15 shows some of the retrieved information in JSON files. The values of "television" and "friends" attributes have changed in week 2 of parsing to include the sub attribute "paging" in addition to "data". This affects the *mAg* when it has to parse the returned JSON file to obtain the list of friends.

Data in Table 7.3 represents information about 15 seed profiles of the mock network. These profiles have been parsed, as described in scenarios 1 and 2, in order to calculate the required time for retrieving 2 JSON files for the seed profiles and their friends. The first file contains all possible information, while the second file contains the profiles' walls.

The average files retrieved are 15 files with an average size of 60 kilobytes. Note

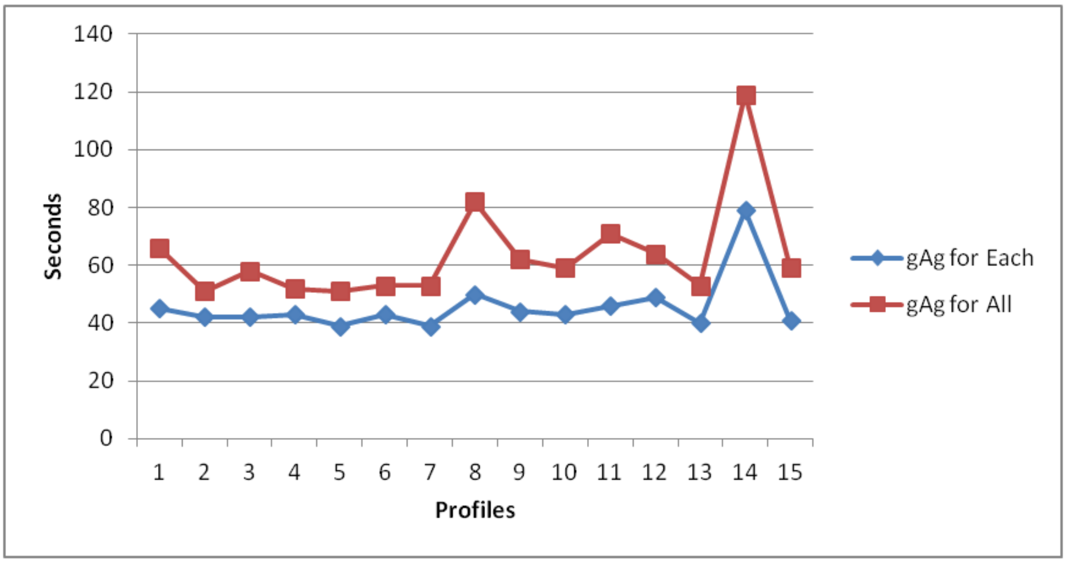


Figure 7.16: Total Retrieval Time of Case Studies 1 and 2

that the port used to transfer files is different from the port used for exchanging messages between agents, to speed up the extraction process.

From the Figure 7.16 a significant difference in the time for retrieving the same amount and size of files can be noticed. The most interesting profile is profile number 14, which has the largest number of friends in the mock network. In case study 1, where one *gAg* is used to retrieve and monitor all profiles, there is a sharp increase in the required time for extracting information compared with case study 2 where one *gAg* is responsible for monitoring each profile.

Moreover, the most time-consuming element in case study 2 is related to allocating *gAg*s and other processing issues. This is supported by the fact that an average of 10 files could be retrieved within 3 seconds. In contrast, each file requires around 1 second to be retrieved in case study 1. Therefore, regarding these results, sub algorithm 2 is better than sub algorithm 1.

In contrast, Table 7.4 illustrates the results of some of the author's real network. The average files retrieved are 220 files with an average size of 15089229 kilobytes.

Seed Profile	No of Friends	Total Files (Unit)	Total Size(KB)	**Total Process (Seconds)**	
				Case Study 1	Case Study 2
1	63	125	7897052	744	279
2	56	105	10580976	815	344
3	294	597	37759190	5185	2142
4	27	55	4119698	734	120

Table 7.4: Some Results of Case Studies 1 and 2 on Real Network

7.8 Summary

This paper continued previous work in developing OSNRS to retrieve information from OSN profiles and monitor the updates in those profiles. OSNRS were improved through the use of API in order to address the drawback of the parser in which it was required to be updated to reflect the changes in the structure of the profile.

Two algorithms were presented aligned with case studies to improve OSNRS. Since the accuracy and correct retrieval and monitoring of OSN profiles is not affected by the size of the sample, the initial results of the experimental work is promising, especially when using sub algorithm 2. Thus, a sub algorithm 3 will be implemented to enhance OSNRS.

However, several improvements to this work could be applied. E.g. when using API, the retrieval process is limited by the allowance of expiry time of the access token. Further investigation in using some features such as offline permission may help to allow OSNRS agents monitoring profiles continuously.

Chapter 8

Conclusions

With the birth of Web 2.0, there has been a significant increase in the number of users involved in generating web contents, and in particular OSNs. Consequently, new methods have to be proposed in order to extract data from these OSNs. What distinguishes OSNs from other web areas is that the developers of mobile devices, smartphones etc. pay more attention to simplifying access to OSN sites, which accordingly speeds up the changes of OSNs in terms of contents as well as structure and representation.

8.1 Research Contribution

The research in this thesis mainly focused on improving existing approaches and studying new proposals for data extraction from web databases, particularly OSN profiles. Before designing the extraction approach, a review was undertaken of three topics which included web data extraction, OSN and MAS. Following this, applications were implemented to extract and monitor the changes in the profiles.

Although there are several studies which have attempted to extract millions of profiles from different OSNs, these studies differ from the work presented in this thesis in two aspects:

- The nature and amount of data extracted are different in terms of that, just profiles' links or some of basic information (e.g. comments and tags) have been extracted. In contrast, the study in this thesis extracts all information provided by the OSN users or OSN providers.

- To the best of the author's knowledge, there is no reported work which monitors the rapid changes in OSN profiles. All previous studies analyzed results gained by visiting each OSN profile only once.

The first phase of the thesis, which was presented in Chapter 4, involved proposing an approach to extract semi-structured and unstructured data from OSNs automatically, by creating a parser with the ability to crawl the source web pages of OSN profiles. The parser was developed from the code presented in [68]. An algorithm was presented and described in detail to implement the "Social Network Extractor" application in order to extract information from Myspace profiles, which was the top OSN at that time, as well as a list of the profiles' top friends and all friends. Some interesting results from the extracted data are shown, accompanied by graphs. A graph is used to visualize the notion of "friendship" in OSNs, where the profiles can be represented as nodes and the relationship between two profiles is symbolized by the edges.

Also, the application has been validated to ensure that the extracted data was accurate. The experiment has shown that:

- Building a parser to extract data from a structured format is easier than from an unstructured or semi-structured format for several reasons. First, the data tags of the web page's source are organized very well. Second, data in a structured format follows the rules strictly. Finally, it is defined in a hierarchical way. Thus, the required tokens can be more easily identified and split.

- The structure of MySpace profiles was found to differ depending on the users' preferences and the type of profile such as public, private, bands and magazines. This proved a challenge when implementing the code. Analysis of HTML structures of various profiles revealed that there was a standard format.

- Even though some of the profiles were private profiles, some attributes, e.g. nickname, gender, age and location, could still be extracted.

- Data that is placed in the repository can be mined and analysed offline to recognize patterns and trends about the social network in which the profiles are based. For example, from the retrieved age and country, it is possible to detect the relationship between the connected profiles in a sub network.

- The profile data can also be used to identify which profile attributes and values make the person vulnerable to social engineering attacks. Vulnerability can be detected by the attributes presented, e.g. if the age and horoscope signs are present on a profile then it is possible to guess roughly when the birthday is. If there are comments present on the profile as well, it may be possible to identify the exact date of the birthday. Other attributes that may contribute to a profile being vulnerable includes whether the individual is a drinker or a smoker.

- The research in this thesis adopted a BFS algorithm to move across the sub network to select the next profiles to be crawled. However, the developed approach could be applied as well to different algorithm, e.g. the Depth First Search (DFS) algorithm if the purpose of extraction requires moving (conditionally) deep in the sub network.

Original results of the work on this subject has been published in [20, 9, 19]

However, the approach developed has experienced two main limitations: firstly, the system is designed as a centralized system, which means that the process of data extraction will stop if the system fails to process one of the profiles, either the seed profile (the first profile specified by the application user to be crawled) or one of its friends. Secondly, the application parser needs to be modified continuously along with updates in the structure of the web page source file.

The second and third phases of the research in the thesis, which were presented in Chapters 5 and 6 respectively, concern overcoming the first limitation. MAS is considered the best solution to overcome the problem of centralization and dependency on fast changes of OSN data formats because of its characteristics, especially in its ability to work in parallel paradigm. Due to the fact that MAS is considered to be a complex system to develop, phase 2 highlights through formal specification (Object-Z) the feasibility of using MAS technology in extracting data from OSN sites in order to ensure that the enhanced web based system, the Online Social Network Retrieval System (OSNRS) is robust, reliable and fits the purpose, before implementing the application. Original work on this subject has been published in [10].

The novelty of the OSNRS lies in having an agent associated with OSN user profile to extract its data and to monitor its updates. The data is sent to a controller agent which saves a history of each user's activities in a local repository. An overview of OSNRS is described informally to illustrate the system, followed by a description of how the information flows between OSNRS components (classes).

Phase 3 implemented the OSNRS application through improvement of the algorithm presented in phase 1 by applying MAS technology, where each agent thread is assigned to a specific profile of an OSN to extract and track data changes. The experimental work of the approach developed is described in detail. In the case study, it was mentioned that two groups of Myspace profiles were selected and observed for two weeks by reactivating the gAgs once a day in order to find out how often users were likely to change their list of top and all friends. Some interesting findings concluded from the experiments include:

- The size of friendship network of a profile mainly depends on two factors:

    1. the number of public profiles in the network.
    2. the size of each public profile's network.

- Analyzing the results of a monitored connected network is more helpful in understanding the behaviour of OSN users and their friends in terms of interaction between the two. Also, the trust between the user and his friends can be valuable in e-commerce.

- From the observation of users' behaviours, it is discovered that users are more likely to change their friends list (adding or removing friends) than they are to change their closer friends in their top list.

Original results of the work on this subject has been published in [8, 13, 11].

The final phase addresses the second limitation of the first phase, which is related to the need for a continuous parser modification to cope with updates in the profile structure. The improved approach involves using API to extract the profile information rather than the parser. The implemented application was applied on a Facebook (which is currently the most used OSN) mock network as well as a real network. The extracted data is saved in a local repository as JSON files. Original results of the work on this subject has been published in [14, 12]

Overall the aims and objectives of the thesis have been achieved through proposing and implementing different algorithms of parsers to extract semi-structured and unstructured data from OSN in case of the source web pages are provided. In the case of the source web pages are unavailable, an algorithm which uses API is presented.

By comparing the two techniques (parsers and API), it is concluded that both of them allow accessing some of basic information even for private profiles. However, each of parser and API has some advantages and disadvantages such that: while the parser allows a direct accessing FOF automatically, API requires the user's authorization and authentication due to it concerns more about privacy. On the other hand, API allows accessing information directly without being affected by the changes in the profile's structure. I.e. DE using API is faster, more precise and accurate in terms of data extracted.

By comparing the two techniques (parsers and API), it is concluded that both of them allow accessing some of basic information even for private profiles. However, each of parser and API has some advantages and dis- advantages such that: while the parser allows a direct accessing FOF automatically, API requires the user's authorization and authentication due to it concerns more about privacy. On the other hand, API allows accessing information directly without being affected by the changes in the profile's structure. I.e. DE using API is faster, more precise and accurate in terms of quality and time of extracting data.

Additionally, some of finding regarding using different algorithms in the thesis are listed below:

- Extracting data using MAS alongside API allows OSNRS agents to obtain the required attributes of an OSN profile despite modifications in the representation of the profile's source web pages.

- Using API is a more ethical way to extract due to the fact that the user has to grant permission to the application in order for its profile and sub network to be extracted (Refer to Appendix D for the ethical statement regarding the work carried out in this phase of research).

- As expected, allocating a gAg for each profile to extract and monitor updates, speeds up the process of extraction even though extra time is added for communication and allocating agents to URLs.
- Saving the extracted data in a JSON file is more efficient than XML files due to its simplicity and speed.

The thesis presents a novel decentralized approach for automated extracting and monitoring data of OSNs. The approach can be applied to any social network as well as many other areas. This research opens up a new direction in the OSN field in terms of monitoring and analysing the behaviour of users and their profiles.

The data extracted in all phases of this study has been analyzed separately by a co-author (Sophia Alim) to be used in her PhD thesis in order to calculate the vulnerability of the friends of the OSN profiles. Some of the results have been published as referenced where appropriate.

8.2 Recommendations for Future Work

Although the research presented in this thesis is promising and positive, there are some limitations in addition to the ones mentioned previously. These limitations include:

- Despite the size of data set extracted being large in some experiments of the thesis regarding Nodexl, it would be considered small in other experiments compared to researchers who have extracted millions of profiles.

- The validation of the experiment has been carried out manually and shows that the extracted data matches the content of the profile. Automated validation through using precision, recall and average precision would help in measuring the quality of data extracted. The precision is used to get the accuracy of the retrieval system while the recall is the percentage of the actual retrieved documents. The higher precision means more probability to retrieve relevant documents while the higher number of the recall implies that the system retrieved the most relevant document containing the required information.

Further to the work reported in this thesis, several advances could be suggested for further research. In terms of expanding the extraction process, it would be recommended to:

- Extract data from another OSN, e.g. Twitter, which is expected to have simpler representation of data formatting (mostly text) in order to analyze data from a more focused group who share common subjects of interest, e.g. the right of women to drive cars in Saudi Arabia. Also, investigating different approaches to extracting data from OSN profiles presented in mobile devices and smartphones.

- Extract data on a larger scale through adding extra workstations to maximize the agent's platform and expedite the extraction process. To enlarge the sample, it is suggested to build a parser based on DOM tree and compare the retrieval and processing time with the parser which is built based on a string tokenizing. DOM tree could be applied on XML documents as well as HTML and XHTML documents. Moreover, DOM tree is tag independent, which means it is faster and easier than string tokenizing.

- Missing information due to privacy issues will need to be factored into studies which analyze information from OSNs. This information may have a major effect on the behaviour of OSN users. Similarly, profiles related to bands and magazines which are excluded from the current study need to be considered in account as well.

- Implement the third scenario presented in Chapter 7 and compare the results with other scenarios in terms of speed and size of data extracted. Note that extra ethical approval is required to allow MYSA application extracting friends of friends' information.
- Use a different agent method which has the ability to communicate through firewalls, e.g. the JXTA agent communication method that is adopted in [101]. The agents of the current OSNRS application cannot communicate through firewalls.

References

[1] Html dom tutorial (2011-12-01).

[2] W3c xml activities (2011-12-01).

[3] Xml dom node tree (2011-12-01).

[4] Knowledge discovery and data mining (2012-02-23).

[5] Definition of : socail network (2012-02-25).

[6] Why, when, and where to use software agents (November 15, 2011).

[7] Abate, A., De Marsico, M., Riccio, D., Tortora, G. Mubai: multiagent biometrics for ambient intelligence. *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–9 (2011).

[8] Abdulrahman, R., Alim, S., Neagu, D., Holton, D., Ridley, M. Multi agent system approach for vulnerability analysis of online social network profiles over time. *International Journal of Knowledge and Web Intelligence*, 280(5360):pp. 98–100 (2012).

[9] Abdulrahman, R., Alim, S., Neagu, D., Ridley, M. Algorithms for data retrieval from online social network graphs. In *10th IEEE International Conference on Computer and Information Technology*, pp. 1660–1666 (2010).

[10] Abdulrahman, R., Holton, D., Neagu, D., Ridley, M. Formal specification of multi agent system for historical information retrieval from online social networks. *Agent and Multi-Agent Systems: Technologies and Applications*, pp. 84–93 (2011).

[11] Abdulrahman, R., Neagu, D. An investigation in using multi agent systems for data retrieval from online social network. In *Proceedings of the 4th Saudi International Conference* (2010).

[12] Abdulrahman, R., Neagu, D., Holton, D. A better way to retrieve information from online social networks. In *Proceedings of the 5th Saudi International Conference* (2011).

[13] Abdulrahman, R., Neagu, D., Holton, D. Multi agent system for historical information retrieval from online social networks. *Agent and Multi-Agent Systems: Technologies and Applications*, pp. 54–63 (2011).

[14] Abdulrahman, R., Neagu, D., Holton, D., Ridley, M., Lan, Y. Data extraction from online social networks using application programming interface in a multi agent system approach. *Transactions on Computational Collective Intelligence*, pp. 84–93 (2012).

[15] Aceto, L., Cimini, M., Ingolfsdottir, A., Reynisson, A., Sigurdarson, S., Sirjani, M. Modelling and simulation of asynchronous real-time systems using timed rebeca. *Arxiv preprint arXiv:1108.0228* (2011).

[16] Acquisti, A., Gross, R. Imagined communities: Awareness, information sharing, and privacy on the facebook. In *Privacy Enhancing Technologies*, pp. 36–58. Springer (2006).

[17] Akerkar, R., Lingras, P. *Building an intelligent Web: theory and practice.* Jones & Bartlett Learning (2008).

[18] Aldea, A., Banares-Alcantara, R., Jimenez, L., Moreno, A., Martinez, J., Riano, D. The scope of application of multi-agent systems in the process industry: three case studies. *Expert Systems with Applications*, 26(1):pp. 39–47 (2004).

[19] Alim, S., Abdul-Rahman, R., Neagu, D., Ridley, M. Data retrieval from online social network profiles for social engineering applications. In *Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for*, pp. 1–5. IEEE (2009).

[20] Alim, S., Abdulrahman, R., Neagu, D., Ridley, M. Online social network profile data extraction for vulnerability analysis. *International Journal of Internet Technology and Secured Transactions*, 3(2):pp. 194–209 (2011).

[21] Amin, M., Jamil, H. Fastwrap: An efficient wrapper for tabular data extraction from the web. In *Information Reuse & Integration, 2009. IRI'09. IEEE International Conference on*, pp. 354–359. IEEE (2009).

[22] Angus, E., Thelwall, M., Stuart, D. General patterns of tag usage among university groups in flickr. *Online information review*, 32(1):pp. 89–101 (2008).

[23] Asfia, M., Pedram, M., Rahmani, A. Main content extraction from detailed web pages. *International Journal of Computer Applications IJCA*, 4(11):pp. 18–21 (2010).

[24] Ashish, N., Knoblock, C. Wrapper generation for semi-structured internet sources. *SIGMOD Record*, 26(4):pp. 8–15 (1997).

[25] Aspray, W., Keil-Slawik, R., Parnas, D. History of software engineering (1996).

[26] Barnett, E. Us election candidates target twitter users (2012-02-26).

[27] Baryannis, G. Formal specification, definitons (21-12-2010).

[28] Baykasoglu, A., Kaplanoglu, V., Erol, R., Sahin, C. A multi-agent framework for load consolidation in logistics. *Transport*, 26(3):pp. 320–328 (2011).

[29] BBC. Facebook timeline: the social network's life story (2012).

[30] Bellifemine, F., Caire, G., Greenwood, D. *Developing multi-agent systems with JADE*, volume 5. Wiley (2007).

[31] Bergman, M. White paper: the deep web: surfacing hidden value. *Journal of Electronic Publishing*, 7(1) (2001).

[32] Bigus, J., Bigus, J. Constructing intelligent agents with java: A programmer's guide to smarter applications (1998).

[33] Bigus, J. P., Bigus, J. *Constructing Intelligent agents with Java, A Programmer's Guide to Smarter Applications*. Wiley (1997).

[34] Blum, T., Keislar, D., Wheaton, J., of Muscle Fish, E. W. Writing a web crawler in the java programming language (2011-12-01).

[35] Bonneau, J., Anderson, J., Danezis, G. Prying data out of a social network. In *Social Network Analysis and Mining, 2009. ASONAM'09. International Conference on Advances in*, pp. 249–254. IEEE (2009).

[36] Bordini, R., Braubach, L., Dastani, M., Seghrouchni, A., Gomez-Sanz, J., Leite, J., OHare, G., Pokahr, A., Ricci, A. A survey of programming languages and platforms for multi-agent systems. *Informatica*, 30:pp. 33–44 (2006).

[37] Bordini, R., Hübner, J., Wooldridge, M. *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. Wiley-Interscience (2008).

[38] Botacci, L., Jones, J. *Formal specification using Z: a modelling approach.* International Thompson Publishing (1995).

[39] Boucherit, A., Khebaba, A., Belala, F. Rewriting logic based approach for the formalization of critical systems based on multi-agent system. *International Journal of Computer Applications*, 13(2) (2011).

[40] Bowen, J. *Formal specification and documentation using Z: A case study approach*, volume 66. International Thomson Computer Press (1996).

[41] Boyd, D., Ellison, N. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13:pp. 210–230 (2008).

[42] Bradshaw, J. Software agents. pp. 1–47 (1997).

[43] Camacho, D., Aler, R., Castro, C., Molina, J. Performance evaluation of zeus, jade, and skeletonagent frameworks. In *Systems, man and cybernetics, 2002 IEEE international conference on*, volume 4, pp. 6–pp. IEEE (2002).

[44] Catanese, S., De Meo, P., Ferrara, E., Fiumara, G., Provetti, A. Crawling facebook for social network analysis purposes. *Arxiv preprint arXiv:1105.6307* (2011).

[45] Caverlee, J., Webb, S. A large-scale study of myspace: Observations and implications for online social networks. *Proc. of ICWSM*, 8 (2008).

[46] Chakrabarti, S., Van den Berg, M., Dom, B. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31(11-16):pp. 1623–1640 (1999).

[47] Chau, D., Pandit, S., Wang, S., Faloutsos, C. Parallel crawling for online social networks. In *Proceedings of the 16th international conference on World Wide Web*, pp. 1283–1284. ACM (2007).

[48] Chlebus, E., Burduk, A., Kowalski, A. Concept of a data exchange agent system for automatic construction of simulation models of manufacturing processes. *Hybrid Artificial Intelligent Systems*, pp. 381–388 (2011).

[49] Cho, J., Garcia-Molina, H. Parallel crawlers. In *Proceedings of the 11th international conference on World Wide Web*, pp. 124–135. ACM (2002).

[50] Crescenzi, V., Mecca, G., Merialdo, P., Missier, P. An automatic data grabber for large web sites. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pp. 1321–1324. VLDB Endowment (2004).

[51] Crestan, E., Pantel, P. Web-scale knowledge extraction from semi-structured tables. In *Proceedings of the 19th international conference on World wide web*, pp. 1081–1082. ACM (2010).

[52] Culotta, A., Bekkerman, R., McCallum, A. Extracting social networks and contact information from email and the web (2004).

[53] Currie, E. *The essence of Z.* Prentice Hall Europe (1999).

[54] Dignum, V., Meyer, J., Dignum, F., Weigand, H. Formal specification of interaction in agent societies. *Formal approaches to agent-based systems*, pp. 37–52 (2003).

[55] Duke, R., Rose, G. *Formal Object Oriented Specification Using Object-Z.* Palgrave Macmillan (2000).

[56] Dwyer, C., Hiltz, S., Passerini, K. Trust and privacy concern within social networking sites: A comparison of facebook and myspace. In *Proceedings of AMCIS*. Citeseer (2007).

[57] Eaglestone, B., Ridley, M. *Web database systems*. McGraw-Hill (2001).

[58] Facebook. Core concepts (2011).

[59] Ferber, J. *Multi-agent systems: an introduction to distributed artificial intelligence*. Addison-Wesley Professional (1999).

[60] Franklin, S., Graesser, A. Is it an agent, or just a program?: A taxonomy for autonomous agents. *Intelligent Agents III Agent Theories, Architectures, and Languages*, pp. 21–35 (1997).

[61] Gaïti, D. *Autonomic networks*. Wiley-ISTE (2008).

[62] Gao, Y., Yuan, F., Zhang, M. Data extraction based on index path in web. In *2010 Second International Workshop on Education Technology and Computer Science*, pp. 157–160. IEEE (2010).

[63] Garcia, A., de Lucena, C., Cowan, D. Agents in object-oriented software engineering. *Software: Practice and Experience*, 34(5):pp. 489–521 (2004).

[64] George, N. Fozcil: A framework for converting formal specifications in object-z to design contracts in oo programming languages (2009).

[65] Ghanea-Hercock, R., Gifford, I. Solutions to security in mobile agent systems. In *IEE seminar on Mobile Agents Where Are They Going?*, p. 5/15/4. IEEExplore (2001).

[66] Gibson, R. Who's really in your top 8: network security in the age of social networking. In *Proceedings of the 35th annual ACM SIGUCCS fall conference*, pp. 131–134. ACM (2007).

[67] Gupta, S., Kaiser, G., Neistadt, D., Grimm, P. Dom-based content extraction of html documents. In *Proceedings of the 12th international conference on World Wide Web*, pp. 207–214. ACM (2003).

[68] Haines, S., Foreword By-Liberty, J. *Java 2 from Scratch with Cdrom.* Que Corp. (1999).

[69] Harry, A. *Formal methods fact file: VDM and Z.* John Wiley & Son Ltd (1996).

[70] Hayes, I., Flinn, B. *Specification case studies.* Prentice-Hall International (1987).

[71] Hayes-Roth, F., Amor, D. *Radical simplicity: transforming computers into ME-centric appliances.* Prentice Hall PTR (2003).

[72] Hedley, Y., Younas, M., James, A., Sanderson, M. Query-related data extraction of hidden web documents. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 558–559. ACM (2004).

[73] Henninger, M. *The hidden web: finding quality information on the net.* Univ of New South Wales (2008).

[74] Hewitt, C. Description and theoretical analysis (using schemata) of planner: A language for proving theorems and manipulating models in a robot. Technical report, DTIC Document (1972).

[75] Hgaret, P. L., Ray Whitmer, L. W. Maintained by the w3c dom ig (2009/01/06).

[76] Hilaire, V., Koukam, A., Gruer, P., Müller, J. Formal specification and prototyping of multi-agent systems. In *Engineering Societies in the Agents World*, pp. 114–127. Springer (2000).

[77] Hinduja, S., Patchin, J. Personal information of adolescents on the internet: A quantitative content analysis of myspace. *Journal of Adolescence*, 31(1):pp. 125–146 (2008).

[78] Hitwise, E. Top 10 social networking websites & forums - august 2011 (2011).

[79] IBM. Api concepts (2012-02-23).

[80] Iglesias, C., Garijo, M., González, J. A survey of agent-oriented methodologies. *framework*, 2:p. 34 (1999).

[81] Jacky, J. *The way of Z: practical programming with formal methods.* Cambridge Univ Pr (1997).

[82] Kalarani, S., Uma, G. Integration of semantic web and knowledge discovery for enhanced information retrivel. *International Journal of Computer Applications IJCA*, 1(1):pp. 112–119 (2010).

[83] Kang, H., Yoo, S., Han, D. Modeling web crawler wrappers to collect user reviews on shopping mall with various hierarchical tree structure. In *Web Information Systems and Mining, 2009. WISM 2009. International Conference on*, pp. 69–73. IEEE (2009).

[84] Kao, H., Lin, S., Ho, J., Chen, M. Mining web informative structures and contents based on entropy analysis. *Knowledge and Data Engineering, IEEE Transactions on*, 16(1):pp. 41–55 (2004).

[85] Knight, J. Knightnet site design - static vs dynamic web pages (2008-07-10).

[86] Kushmerick, N. *Wrapper Induction for Information Extraction*. Ph.D. thesis, University of Washington (1997).

[87] Laender, A., Ribeiro-Neto, B., Da Silva, A., Teixeira, J. A brief survey of web data extraction tools. *ACM Sigmod Record*, 31(2):pp. 84–93 (2002).

[88] Lange, D., Oshima, M., Mishuru, O. Programming and deploying java mobile agents with aglets (1998).

[89] Lawrence, S., Giles, C. Searching the world wide web. *Science*, 280(5360):pp. 98–100 (1998).

[90] Lenhart, A., Madden, M. Social networking websites and teens: An overview. *Pew Internet & American Life Project*, 3 (2007).

[91] Levene, M. *An introduction to search engines and Web navigation*. Wiley.

[92] Liu, B., Zhai, Y. Net–a system for extracting web data from flat and nested data records. *Web Information Systems Engineering–WISE 2005*, pp. 487–495 (2005).

[93] Luck, M., Griffiths, N., d'Inverno, M. From agent theory to agent construction: A case study. *Intelligent Agents III Agent Theories, Architectures, and Languages*, pp. 49–63 (1997).

[94] Ludwig, S., Van Santen, P. A grid service discovery matchmaker based on ontology description. In *Proceedings of the International EuroWeb 2002 Conference, Oxford, UK* (2002).

[95] Ma, L., Goharian, N., Chowdhury, A. Automatic data extraction from template generated web pages. In *PDPTA'03*, pp. 642–648 (2003).

[96] Madhavan, J., Ko, D., Kot, Ł., Ganapathy, V., Rasmussen, A., Halevy, A. Google's deep web crawl. *Proceedings of the VLDB Endowment*, 1(2):pp. 1241–1252 (2008).

[97] Mahmoud, Q., Yu, L. Making software agents user-friendly. *COMPUTER-IEEE COMPUTER SOCIETY-*, 39(7):p. 95 (2006).

[98] Martinez-Sarriegui, I., Zhu, H., Shan, L., García-Sáez, G., Gómez, E., Hernando, M. Modeling and formal specification of a multi agent telemedicine system for diabetes care. In *Proceedings of the International Conference on Agents and Artificial Intelligence, ICAART2009*, pp. 507–512 (2009).

[99] Meghabghab, G., Kandel, A. *Search engines, link analysis, and user's Web behavior*, volume 99. Springer-Verlag New York Inc (2008).

[100] Mika, P. Flink: Semantic web technology for the extraction and analysis of social networks. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):pp. 211–223 (2005).

[101] Milani Fard, A., Kahani, M., et al. Multi-agent data fusion architecture for intelligent web information retrieval. *International Journal of Intelligent Systems and Technologies*, 2 (2007).

[102] Mislove, A., Marcon, M., Gummadi, K., Druschel, P., Bhattacharjee, B. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pp. 29–42. ACM (2007).

[103] Mohammadian, M. *Intelligent agents for data mining and information retrieval.* Idea Group Publishing (2004).

[104] Mohammadian, M., Jentzsch, R. Computational intelligence techniques driven intelligent agents for web data mining and information retrieval. *Intelligent agents for data mining and information retrieval*, pp. 15–29 (2004).

[105] Munoz, A., Anton, P., Mana, A. Static mutual approach for protecting mobile agent. In *International Symposium on Distributed Computing and Artificial Intelligence*, pp. 51–58. Springer (2011).

[106] Niazi, M., Hussain, A. A novel agent-based simulation framework for sensing in complex adaptive environments. *Sensors Journal, IEEE*, 11(2):pp. 404–412 (2011).

[107] Ning, K., Yang, R. Mas based embedded control system design method and a robot development paradigm. *Mechatronics*, 16(6):pp. 309–321 (2006).

[108] Nwana, H. S. Software agents: An overview. *Knowledge Engineering Review*, 11:pp. 205–244 (1996).

[109] Orenstein, D. Quickstudy: Application programming interface (api) (2000).

[110] Pandit, S., Chau, D., Wang, S., Faloutsos, C. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, pp. 201–210. ACM (2007).

[111] Park, J., Barbosa, D. Adaptive record extraction from web pages. In *Proceedings of the 16th international conference on World Wide Web*, pp. 1335–1336. ACM (2007).

[112] Potter, B., Till, D., Sinclair, J. *An introduction to formal specification and Z.* Prentice Hall PTR (1996).

[113] Qian, S. Natural language processing applications. lecture 8: Information extraction -1 (2011-12-01).

[114] Quah, J., Chen, Y., Leow, W. Networking e-learning hosts using mobile agents. *Intelligent agents for data mining and information retrieval*, p. 262 (2004).

[115] Quah, J., Leow, W., Chen, Y. Mobile agent assisted e-learning. In *First International Conference on Information Technology & Applications (ICITA 2002), Bathhurst, Australia* (2002).

[116] Ratcliff, B. *Introducing Software Engineering Specification Using Z: A Practical Case Study Approach.* McGraw-Hill, Inc. (1994).

[117] Ricca, F., Tonella, P. Using clustering to support the migration from static to dynamic web pages. In *Program Comprehension, 2003. 11th IEEE International Workshop on*, pp. 207–216. IEEE (2003).

[118] Ricci, A., Bordini, R., Agha, G. Agere!(actors and agents reloaded): splash 2011 workshop on programming systems, languages and applications based on actors, agents and decentralized control. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, pp. 325–326. ACM (2011).

[119] Saaman, C., Saaman, E., Klint, P., Mosses, P., Hesselink, W. Another formal specification language (2000).

[120] Schildt, H., Holmes, J. *The art of java.* McGraw-Hill Osborne Media (2003).

[121] Sherman, C., Price, G. *The invisible Web: Uncovering information sources search engines can't see.* Information Today, Inc. (2001).

[122] Sherman, C., Price, G. *The invisible web.* CyberAge Books (2002).

[123] Srinivasanacd, P., Mitchellab, J., Bodenreidera, O., Pantc, G., Menczerc, F. Web crawling agents for retrieving biomedical information (2001).

[124] Strater, K., Richter, H. Examining privacy and disclosure in a social networking community. In *Proceedings of the 3rd symposium on Usable privacy and security*, pp. 157–158. ACM (2007).

[125] Sygkouna, I., Anagnostou, M. Efficient information retrieval using mobile agents. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 1241–1242. ACM (2005).

[126] Trusov, M., Bucklin, R., Pauwels, K. Effects of word-of-mouth versus traditional marketing: Findings from an internet social networking site. *Journal of Marketing*, 73(5):pp. 90–102 (2009).

[127] university of the pacific. Online social networking dangers and benefits (2012-02-26).

[128] Viswanath, B., Mislove, A., Cha, M., Gummadi, K. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*, pp. 37–42. ACM (2009).

[129] Wooldridge, M. *An introduction to multiagent systems.* Wiley (2009).

[130] Wooldridge, M., Jennings, N. Intelligent agents: Theory and practice. *Knowledge engineering review*, 10(2):pp. 115–152 (1995).

[131] Yves, C., Jeannot, E. New dynamic heuristics in the client-agent-server model. In *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pp. 11–pp. IEEE (2003).

[132] Zhu, H. A formal specification language for mas engineering. In *The 2nd International Workshop on Agent-Oriented Software Engineering* (2001).

[133] Zhu, H. Formal specification of agent behaviour through environment scenarios. *Formal Approaches to Agent-Based Systems*, pp. 263–277 (2001).

[134] Zhu, H. A formal specification language for agent-oriented software engineering. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 1174–1175. ACM (2003).