Addis Ababa University

College of Natural Sciences

Design of a Dark Web Crawler and Offline Language Identifier for Amharic Documents

Daniel Adenew Wondyifraw

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia

February 2016

Addis Ababa University

College of Natural Sciences

Daniel Adenew Wondyifraw

Advisor: Mesfin Kifle *(PhD)*

This is to certify that the thesis prepared by *Daniel Adenew Wondyifraw*, titled *Design of a Dark Web Crawler and Offline Language Identifier for Amharic Documents* and Submitted in partial fulfilment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by Examining Committee:

	Name	Signature	Date
Advisor:	*Mesfin Kifle (PhD)*	_____	_____
Examiner	_____	_____	_____
Examiner:	_____	_____	_____

Abstract

Document searching over the Internet has become daily practice of people for their personal and business matters. Though, there are billions of websites easily available, still many more are not easily accessible. In contrary with surface web or Clearnet, a content inside TOR network which require specific software, configurations or authorization to access from the public Internet commonly referred as DarkWeb. Due to this fact Clearnet search engines like Google are not capable of searching its content.

In contrary to the name "DarkWeb", DarkWeb contains a collection of useful and legal information that can be used for our day to day activity. In fact, its darkness refers the content is being hidden from the Clearnet search engines. As a result, we proposed a design of a Dark Web crawler to discover and give an insight for the DarkNet content, especially for TOR network. We also proposed integration of a language identifier component to be used to identify Amharic content. These were the gaps seen on related researches on DarkWeb regarding crawling and content analysis. These researches were conducted in small dataset for only specific types of DarkWeb sites and did not considered contents available in Amharic language.

The main objective of this thesis is designing an architecture for Dark Web crawler for TOR network. Basically, the proposed architecture is composed of a recursive light-weight crawler threads using a Fork-Join parallelism, a concurrent persistence storage manager with a persistent media access, URL queue for tracking links, a Download manager to download dark web contents, and HTML texts are compressed using an HTML compressor and language identification with an offline language identifier components.

In the proposed system a Java programming language is used to develop the prototype referred as Dark Web crawler. We have tested the performances of our proposed design using the downloaded web documents and the crawled information. We have collected over 13,000 hidden services and 67,602 dark web URLs and downloaded 56,304 DarkNet web sites, resulting 800 MB data. Google search engine is used to evaluate results for selected number of datasets using parameters (*i.e.,* page title and meta-tag). Out of the selected 30 data sets 7 are Amharic. We found promising result using the proposed system in finding all whereas a Google search engine were not able to find any of them from top 10 returned results.

Key Words: - *Dark Web, Dark Net, Dark Wet crawler, DarkNet Crawler, Fork-Join, Hidden Services, Google, .Onion, TOR*

Dedicated

To

My Grandfather

Wondyifraw Ejegu Bereseaw

Acknowledgments

First and foremost, I would like to God and His Holy Mother St Marry, the Seven Arc Angles and Priests in Heaven, for I believe in them and gave me a strength, hope, determination and wisdom to bring this thesis to completion.

"The LORD is my strength and my shield; my heart trusts in him, and he helps me"

Second, my gratitude goes to my advisor Dr. Mesfin Kifle for his ideas, experiences, and discussions with me throughout the course of writing this thesis. I salute you, Sir!

Third, this thesis would not have been possible without the support of my family, especially to my loving Mother Yiftusera Kassahun, to my Father Adenew Wondyifraw and my Sister Mekedes Adenew, I love you so much!

Last but not least, my deepest gratitude to Mr. Craig Robinson!

May God Bless You! I wish you all a Happy Long Live!

Table of Contents

LIST OF TABLES IV

LIST OF FIGURES V

LIST OF ALGORITHMS VI

LIST OF ACRONYMS VII

CHAPTER ONE: INTRODUCTION 1

1.1 BACKGROUND 1

1.2 MOTIVATION 2

1.3 STATEMENT OF THE PROBLEM 2

1.4 OBJECTIVES 3

1.5 METHODS 4

1.6 APPLICATION OF RESULTS 4

1.7 SCOPE AND LIMITATIONS 5

1.8 ORGANIZATION OF THE REST OF THE THESIS 6

CHAPTER TWO: LITERATURE REVIEW 7

2.1 INTRODUCTION 7

2.2 INFORMATION RETRIEVAL 7

2.3 WEB SEARCH ENGINES 9

2.3.1 Components of Search Engine 11

2.3.2 Web Crawler 12

2.3.3 Data Miners 16

2.4 THE DEEP WEB VS. DARK WEB 18

2.5 TOR NETWORK 19

2.6 DATA ANALYSIS ON TOR 25

2.7 AMHARIC LANGUAGE 26

2.8 LANGUAGE IDENTIFICATION OF WEB PAGES BASED ON IMPROVED N-GRAM ALGORITHM 27

2.9 SUMMARY 28

CHAPTER THREE: RELATED WORK 29

3.1 INTRODUCTION 29
3.2 AUTOMATIC CONTENT ANALYSIS OF TOR NETWORK 29
3.3 SUMMARY 33
CHAPTER FOUR: THE PROPOSED SYSTEM 35
4.1 INTRODUCTION 35
4.2 DESIGN CONSIDERATION 35
4.3 THE PROPOSED ARCHITECTURE 35
4.3.1 URL Queue 36
4.3.2 Fork Join Thread Manager 36
4.3.1 Link Extraction Module 37
4.3.2 TOR Controller 41
4.3.3 Concurrent Persistence Storage Manager 42
4.3.4 Download Manager 43
4.3.5 HTML Compressor 45
4.3.6 Offline Language Identifier 46
4.3.7 TOR Client and DNS Lookup 48
4.4 SUMMARY 48
CHAPTER FIVE: EXPERIMENT 49
5.1 INTRODUCTION 49
5.2 DATA SETS 49
5.3 IMPLEMENTATION 51
5.4 TEST RESULT 60
5.5 DISCUSSION 69
CHAPTER SIX: CONCLUSION AND FUTURE WORK 71
6.1 CONCLUSION 71
6.2 CONTRIBUTION OF THIS WORK 72
6.3 FUTURE WORK 73
REFERENCES 74

ANNEX 1: DARK WEB GOSPEL MINISTRIES WEBSITE.....80

ANNEX 2: DARK WEB SITE IN AMHARIC LANGUAGE.....81

ANNEX 3: DARK WEB LIVE RADIO NEWS WEBSITE 82

ANNEX 4: DARK WEB SOCIAL NETWORKING AND MUSIC.....83

ANNEX 5: DARK WEB WORLD HISTORY ARCHIVES.....84

List of Tables

Table 5.1: *Data Set Used* 50

Table 5.2*: Test result of the designed Dark Web Crawler* 61

Table 5.3*: Test Result from Google Search Engine Crawler* 63

Table 5.4 : *Summary of results from Dark Web Crawler* 69

List of Figures

Figure 2.1: *High Level Architecture of a Search Engine* 10

Figure 2.2: *Search Engine Components* 11

Figure 2.3: *Crawler Architecture* 13

Figure 2.4: *Flow of communication on TOR network* 22

Figure 2.5: *The Deepest and Darkest Corner of the Internet* 23

Figure 2.6: *Flowchart for Language Identification Process* 28

Figure 4.1: *Architecture of a Dark Web Crawler and Offline Language Identifier* 39

Figure 4.2: *The modified architecture of the Offline Language Identifier using G2LI* 47

Figure 5.1: *Running Prototype for Dark Web Crawler* 55

Figure 5.2: *Download Manager and HTML Compressor User Interface* 56

Figure 5.3: *Conceptual Database Schema Definition of a Dark Web Crawler and Offline Language Identifier* 57

Figure 5.4 *: Sample Dark Net Website Downloaded* 58

Figure 5.5: *Running Prototype for Language Identifier* 59

List of Algorithms

Algorithm 4.1: *The Proposed Fork-Join Based Dark Web Crawler* 38

Algorithm 4.2*: Link Extractor* 40

Algorithm 4.3: *Dark Web Address Filter Component* 41

Algorithm 4.4: *Concurrent Persistent Storage Manager* 43

Algorithm 4.5: *Algorithm for Download Manager* 45

List of Acronyms

C&C	Command and Control
DHT	Distributed Hash Table
DNS	Domain Name System
DoS	Denial-of-Service
G2LI	Global Information Infrastructure Laboratory Language Identifier
I2P	Invisible Internet Project
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IR	Information retrieval
JDK	Java Development Kit
JSP	Java Server Page
LDA	Latent Dirichlet Allocation
NAT	Network Address Translation
OR	Onion Route/r
P2P	Peer-to-Peer
PHP	Pre Hypertext Processor
SMTP	Simple Mail Transfer Protocol
TLD	Top Level Domain
TOR	The Onion Route Protocol
UI	User Interface
WWWW	World Wide Web Worm

Chapter One: Introduction

1.1 Background

In today's world with the spontaneous growth of technology, innovation and the Internet we are living in the era of information. Information plays a vital role in our day to day activity. Since, the innovation of Internet brought to public use in 1960's the technology has grown enormously in past few decades. It is believed to be having over 3 billion users in 2015 [1]. About 46.4% of the world's population uses the Web for news, entertainment, and communication and myriad other purposes [2]. As the evolvement of digital engagement increases, users as well as mobile and computer devices were posed with a privacy concern, the right of anonymity, electronic fraud or theft and in general security related attacks. Anonymity is not something which was invented with the Internet. Anonymity is often used to protect the privacy of people. Allowing the people not only to access information anonymously but also to publish anonymously is an important aspect nurturing democracy.

The Onion routing is one of the established techniques to provide sender- or receiver-anonymity invented by Reed *et al.* [3]. It is especially applicable for building low-latency anonymous communication systems e.g., web browsing. Users can preserve their anonymity online and make their content inaccessible through the Internet, using networks that provide anonymity. The most common of these are TOR (originally The Onion Router) [4] and the Invisible Internet Project (I2P) [5]. TOR also known as Third-generation Onion Router uses Onion Route as default protocol for routing and it was developed in the mid-1990s by the U.S. Naval Research Laboratory with the purpose of protecting U.S. intelligence communications online. TOR is a volunteer-based anonymity network consisting of over 3000 relay servers [6]. The network provides privacy to users accessing Internet services, but it also provides a way to anonymously make TCP services available as so-called hidden services. TOR is based on the idea of onion routing. In a nutshell this means that the data which is sent over the network is first packed in multiple layers of encryption, which are peeled off one by one by each relay on the randomly selected route the package travels [7].

To differentiate between the regular Internet and Darknet, the term "Clearnet" refers to the normal, publically accessible Internet at large. Darknets' function inside the Internet, and

are dependent on it to function, but their design theories and goals dictate that they operate under fundamentally different principles than that of the Clearnet. When a Clearnet user at home opens a website (for e.g. *ww.aau.edu.et)* in his/her web browser, data is first sent to the nearest router. This is often forwarded to their ISP's routing infrastructure and it continues until the last hop, the server that hosts the requested page (Ethio Telecom .*et* domain) is reached. All data transmitted through this method is unencrypted, allowing for interception, data alteration or observation. When a user of Darknet accesses a website, data is first sent to the TOR routing software installed on the user's machine. This software produces encrypted tunnels to one or more TOR entry relays. Data is transmitted from the TOR software through a tunnel to the TOR entry relay, which has itself encrypted tunnels to other TOR relays. Data is forwarded to a various other TOR relays, before it arrives at a TOR exit point or exit node, which is an exit point from TOR into the Clearnet. From here onwards data will traverse Clearnet paths to the server, which then services the request. The actual path that the data took inside TOR cannot be predicted, but only the Clearnet route, that the data took into TOR and the route it took when it exited TOR can be seen. All data sent via TOR is encrypted in such a way that the original data source and destination cannot be predicted, and the data cannot be changed.

1.2 Motivation

Searching online information is the most frequent and often performed activity commonly assisted by search engines. Search engines use software called crawlers to scan web content in a fashioned manner. However, there are a number of cases they are not able to scan the entire web and in many cases they fail to retrieve content accessible from the web. Current search engines like Google are only capable of crawling, indexing and searching, content and website that are only available on World Wide Web. After crawling of individuals dark web documents there is a need for identification of the language of each web documents, particularly for Amharic contents. It is, therefore, the need for conducting this work to come up with a system that provides language identification of Amharic documents. This will help to know the availability of how much local content? It helps to identify the gap in the content analysis of local language based content inside TOR network.

1.3 Statement of the Problem

The surface web which people use often, consists of data that search engines can find and show up based on user query [8]. According to a research by Bergman [9] argues that

Clearnet search engines can reach only 0.03 percent of the information that is available. Bergman [9] claim that much of the rest content were submerged in what is called the Deep Web also known as the “Dark Net”, “Invisible Web”, “Dark Web “and “Hidden Web”. TOR is a popular ‘dark net’, a network that aims to conceal its users’ identities and online activities. This dark nets is composed of host machines that cannot be accessed by conventional means, which is why the content hosted is typically not indexed by traditional search engines like Google and Bing [7].

Crawling the full collection of onion addresses is prevented by several objective reasons: firstly, many hidden service operators are not interested in their existence to be widely known. Furthermore, hidden services or websites on TOR network only rarely link to each other – which hold back traditional crawling [10]. For this reason standard surface search engines were faced with a challenge.

In contrary to what the name “DarkWeb” give to the DarkNet websites, contains useful and legal information that can used for our day to day actives. In fact the darkness refers the content is being hidden from the Clearnet search engines. To the best of the researcher’s knowledge surface search engine does not crawler TOR network. Therefore, we proposed to fill this gap by designing an architectural model of a Dark Web crawler incorporated with an offline language identifier for Amharic content.

1.4 Objectives

General Objective

The general objective of this thesis is to design and implement architectural model of a TOR search engine crawler for crawling DarkWeb and identify Amharic language content.

Specific Objectives

The specific objectives are:

- Review literatures with regard to search engines , crawlers , content analysis , language identification , TOR and Information Retrieval
- Study the hidden services (DarkWeb) of TOR with respect to crawling specifics.
- Study language identification techniques focusing Amharic language
- Design the architecture of Dark Web crawler.
- Design algorithm for crawling Dark Web and develop the prototype.
- Collect a data set for test and evaluation purpose content identification & crawling

- Evaluate the proposed DarkWeb crawler findings with Google search engine.

1.5 Methods

In order to conduct this research work, the methods mentioned below will be used to select and implement appropriate methods and techniques.

Literature Review

A number of research papers and publication will be examined and reviewed on different areas like crawling techniques and algorithms, TOR network, process parallelism approach, content analysis, traffic analysis, security and web crawling.

Crawling

The crawling will carried out using a parallel thread of crawlers. A fork-join along with a breadth first algorithm will be used to design the crawler.

Data collection

Date set is going to be collected from dark websites found inside TOR Network that can be used for test and evaluation purpose later on.

Tools

Java programming language will be used to implement the selected crawling techniques and to develop modules involved in crawling process and User interface for the individual proposed prototypes.

MySQL Database will be used to persistent data about crawling and identified language data from a language identifier tool (*i.e.*, G2LI) modified to be used from a UI (user interface), and save identification results into a database.

1.6 Application of Results

Web crawler has many applications. But, much has not been done; regarding the sort of thing on crawling DarkWeb and TOR network. The probable application of this research is significant which benefit a many stockholders directly and indirectly. At front, scientific researchers will be benefited from the model of the architecture and the prototype designed and developed, since there was no such contribution in the area. Also, online user gain advantages in searching content and finding resources currently not available to standard search engine crawlers.

The proposed system can also serve as data mining tool for researches or data consumers when looking for using in data mining activities. Different stakeholders and journalist will find information about censored or archived information since TOR network allows anyone with a freedom of speech and content to be hosted on the network ., even banned website from the World Wide Web (e.g., WikiLeaks).

Users can also extend their search results by including a dark web contents *i.e.*, hidden. DarkWeb also contains similar content like the Clearnet that are hosted and served to public interest. This include anonymous email services, useful blogs in English and Amharic languages, music and film downloads, archived information, chats, social networks and etc. User also gain additional privacy in accessing and publishing content anonymously

Government will have insight into the activities that take place in dark net like TOR which may be very useful for different reason, due to its anonymity. For instance, monitoring activities is possible when having the right perspective by looking and analyzing information gained out of the proposed crawler into further investigations.

Furthermore, this study will show the gap between DarkWeb crawlers for TOR network against surface search engine crawlers. The discovered DarkWeb URLs and their structural parameters recorded into to database when crawling can serve as data set for future researchers along with the downloaded DarkWeb websites to determine linguistic behaviors in DarkNet for language content like Amharic. During crawling activity figuring out vurnabaility, security holes on the network can contribute security feedback to TOR project in the protection of anonymity as well adds value in the future security researches.

1.7 Scope and Limitations

Building a search engine crawler is a complex task. The fact that algorithm used for crawling often require sound hardware resources mainly Memory, Storage and CPU with multithreading and concurrency control. DarkWeb contains several heterogeneous Web documents such as image, audio, video, pdf, zip, excel and text. However, this research work assumes a text based indexing and download of contents. This study will also be limited to crawler and identify Amharic and English based web contents.

1.8 Organization of the Rest of the Thesis

Organization of the thesis is described next. Chapter Two covers detail discussion of what deep web is? How deep web is related to DarkNet? What are Hidden Services? And How TOR works? It gives an insight into methods of Information Retrieval, type of search engine and crawling strategies. Chapter Two also deals with the type of content analysis on TOR network and overview of Amharic language. Chapter Three tries to show efforts of different researchers in automatic content analysis of the TOR Network. Each of the works presented with its pros and cons. The Fourth Chapter broadly explains and discusses the design of the proposed Dark Web crawler and Offline Language Identifier for Amharic documents. The proposed system's architecture components are also covered in separate section, in chapter four. Finally, Chapter Five explains about our implementations, experimentations and summarizes our findings. Finally, conclusion and recommendation is given in Chapter Six along with future works. In addition, references and appendixes are given at the end.

Chapter Two: Literature Review

2.1 Introduction

In this chapter, we try to provide an overview of the domain knowledge concepts related to TOR network, information retrieval and crawlers, anonymity and privacy and language identification and Amharic language is presented. We believe that providing explanatory concepts on this regard is necessary to aid our reader for clarifying used knowledge on our study. The size of the World Wide Web has grown enormously in the past few decades. It has become huge in amount and diverse on its content. As a matter of fact, the increasing sizes of the WWW lead to invention of looking for better information retrieval means. Information retrieval concepts will be discussed on the first section of this chapter. Search engine is one of the invented IR system a tool is used to search content which usually incorporates different software components.

Crawler are the main building blocks of a search engine. Crawler help a search engine by traversing websites following in-link and out-link in fashioned and recursive manner. The algorithms, strategies and techniques for modeling crawler are going to be discussed. Since, this study focuses on model of a crawler component for TOR Network also known as “Dark web “ and also a part of the “Deep web” , the concept related to Deep web, Dark web and TOR network will be covered in brief and precise details separate sections. Finally, the concept related to CPU scheduling, muli-threading and fork-join algorithm that is used to carry out recursive task like web crawling will be discussed.

2.2 Information Retrieval

The meaning of information retrieval can be very broad. But according to [11], it is defined as follows:

“Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).”

Information Retrieval (IR) is the discipline that deals with retrieval of unstructured data, especially textual documents, in response to a query or topic statement, which may itself be unstructured, e.g., a sentence or even another document, or which may be structured, e.g., a Boolean expression [12].

Approaches of Information Retrieval

Generally, there are two major categories of IR technology and research. These are Semantic and Statistical. Semantic approaches attempt to implement some degree of syntactic and semantic analysis; In other words, they try to reproduce to some (perhaps modest) degree the understanding of the natural language text that a human user would provide [12]. In statistical approaches, the documents that are retrieved or that are highly ranked are those that match the query most closely in terms of some statistical measure. There are different models in statistical approaches; including Boolean model, Extended Boolean model, Vector space model and Probabilistic model. Most common to all statistical approaches query terms are collected from the entire text document and measured statistically. Then the terms or words usually undergo into preprocessing operation. Preprocessing operation like stemming, stop word removal and root form formation words for each word is applied. This improve efficiency, accuracy and also eliminate the ambiguity that arises from the occurrence of different grammatical forms of the same word, e.g., "research," "researched," "researches," and "researching" should all be recognized as forms of the same word.

Evaluation of IR Performance

At the heart of IR evaluation is the concept of "relevance". Relevance is an inherently subjective concept in the sense that satisfaction of human needs is the ultimate goal, and hence the judgment of human users as to how well retrieved documents satisfy their needs is the ultimate criterion of relevance. Relevance depends not only on the query and the collection but also on the context, e.g., the user's personal needs, preferences, knowledge, expertise, language, etc. [12].

Two measures of IR success, both based on the concept of relevance (to a given query or information need), are widely used: "precision" and "recall" calculated by Equations 1 and 2 respectively.

Precision (P) is the fraction of retrieved documents that are relevant [12].

$$Precision = \frac{\#(relevant\ items\ retrived)}{\#(retrived\ items)} \qquad (1)$$

Recall (R) is the fraction of relevant documents that are retrieved [12].

$$Recall = \#\frac{(relevant\ items\ reterived)}{\#(relevant\ items)} \qquad (2)$$

2.3 Web Search Engines

Before Web search engines introduced users use large and widespread system of the World-Wide Web and find resources by following hypertext links from one document to another. WWWW (World Wide Web Worm) and WebCrawler, the Web's first comprehensive full-text search engine, were invented and evolved subsequently from 1994 to 1997, which helped fuel the Web's growth by creating a new way of navigating hypertext: searching [13].

An increasing problem of the World Wide Web is enormous number of resources available and the difficulty of locating and tracking everything. Nowadays, the Web has grown enormously and is encompassing more than billions of information databases and huge heterogeneity of contents. Therefore, thinking only to navigate through hyperlinks on HTML web pages is difficult. Due to the rapid increasing size of WWW, search engines are becoming increasingly important as the primary means of finding relevant information.

Search engines are special software tools that are designed to help people find information stored on the Web. Search engines have their roots in IR systems, which prepare a keyword index for the given corpus and respond to keyword queries with a ranked list of documents. The query language provided by most search engines return results of Web pages and different type documents that contain (or do not contain) specified words and phrases [14]. Such search engines rely on massive collections of web pages that are acquired with the help of a web crawler, which traverse the web by following hyperlinks and storing downloaded pages in a large database that will be indexed, ranked and polished for efficient execution of user query term hit within document content [15] .

According to a research [16] conducted in 2014 a web search is currently generating more than 13% of the traffic to Web site. In today's world search engines like Google is most known and also search engines like Bing, Yahoo and AltaVista are commonly used. Even though, there are differences in the way these various search engines work the task performed can be generalized into four common steps as follows:

1. Search engines automatically crawl and download Web pages using web crawler.

2. Search engines use Indexer to perform indexing and store the inverted index from the downloaded Web content into repositories (Link and Document Repositories).
3. Search engines perform content relevance ranking and rating using ranking algorithm (Scoring).
4. Search engines, provide the end user refined results based on query word or combinations of words referenced with preexisting index by Query Engine.

The organization of a search can be described as having a thread of crawlers – with a main crawler controllers and manages individual worker threads, a link repository which serve as URL frontier to keep track of visited and non-visited URLs, later Document repository to store downloaded web pages for offline Information Retrieval operation like indexing – using an Inverted Index of websites or documents. Later, scoring of result (*i.e.*, Precision and Recall) will be calculated using user‘s query terms over downloaded and indexed web documents. The user’s query is feed into the system using a Query engine and search result are then returned for matching documents. The High-level architecture of a search engine and the tasks performed is shown in Figure 2.1 [17].

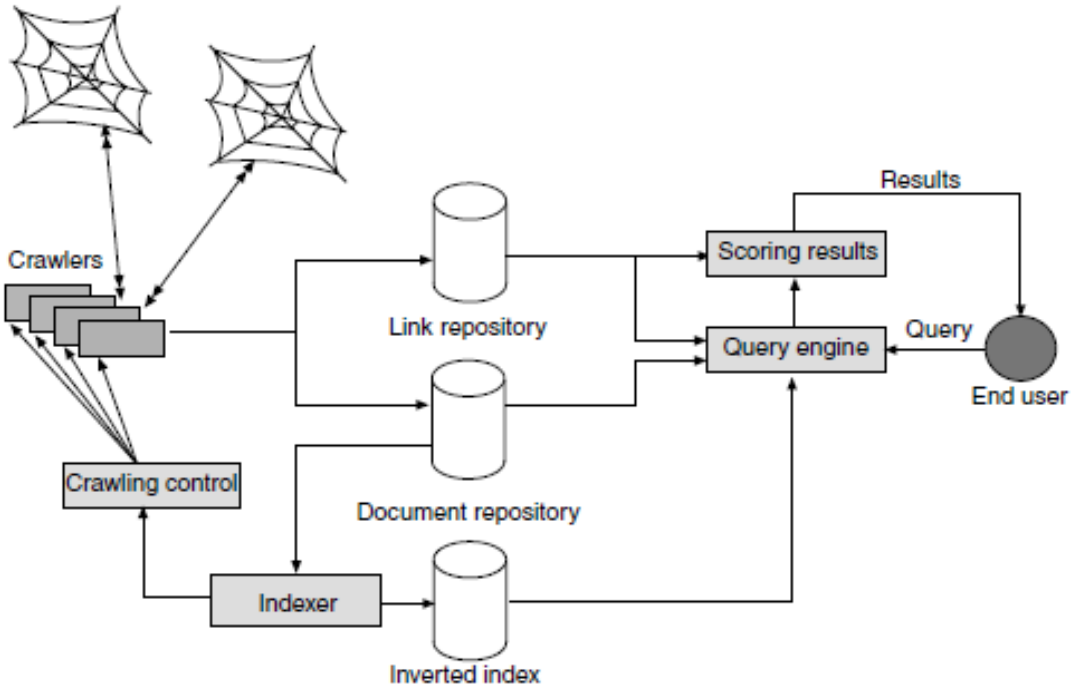


Figure 2.1: *High Level Architecture of a Search Engine*

2.3.1 Components of Search Engine

Search engine is made of different modular entities glued together to work as single coherent system. To accomplish the intended purpose of searching and retrieval a search engine must functionally collaborate with its modular components. Search engines consist of two fundamental components web crawlers, which find, download, and parse content in the WWW (World Wide Web), and data miners, which extract keywords from pages, rank document importance, and answer user queries. These components are the building block of any search engine and depending on the type of the search engine. There might exists a slight adjustment in organization. But, the following are main components of a search engine:

1. Web Crawlers (Spider or Bot)
2. Data Miners (Indexer, Page Rank and Query Processor)

Crawlers are often involved in traversing using the extracted in and out links in a given web page. Indexer is used to index word to represent the web page using systematic manner by keeping only relevance details of the downloaded web document and later a query engine or query process serve as a gateway between the search system and users querying the search engine. The major components of a search engine are shown in Figure 2.2 [18].

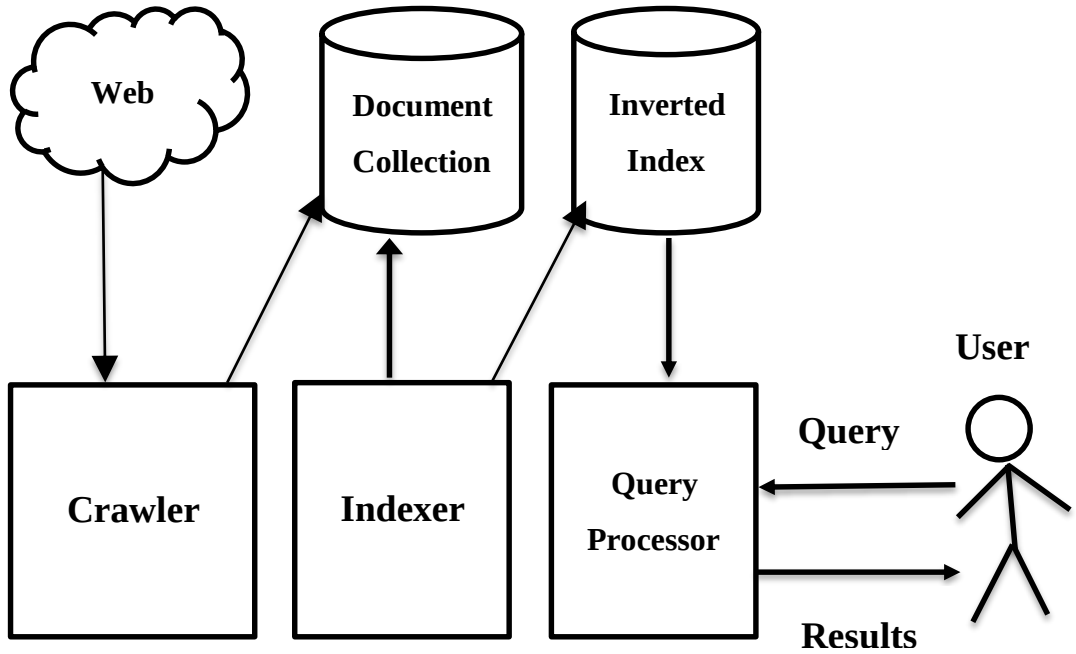


Figure 2.2: *Search Engine Components*

2.3.2 Web Crawler

The web crawler which is sometimes referred to spider, bot or agent is a computer program that browses the web in recursive and automated manner. A crawler is the most profound component of a search engine which helps the search engine to download content and resource from the web or any given network. The objective of crawling is to quickly and efficiently gather as many useful web pages as possible, together with the link structure that interconnects them. Crawling is a recursive process and most of the time it might not halt. The process generally starts with a set of Uniform Resource Locator (URLs) called the Seed URLs. Often crawler follows the links found on web pages and they collect the in-links and out-links recursively till all links get visited. The seed URL (the starting point for the crawling process) can be any arbitrary URL, but when selection of seed URL should be wise. Since, not selecting a seed URL that be used to traverse additional links might halt the crawling process too soon without reaching the goal for, it is always advisable to select a good seed URL that contains many in-links and out-links [19].

Despite the numerous applications for web crawlers, at the core they are all fundamentally the same. Following is the process by which web crawler's work:

- ✓ Download the Web page
- ✓ Parse through the downloaded page and retrieve all the links.
- ✓ For each link retrieved, repeat the process.

At first attempt, implementation of the crawling system may appear trivial. This is however not true in the high performance and industry strength crawlers the case where several hundred or even a thousand pages have to be downloaded per second. The freshness, newness and revisiting of a page also has a significant importance while crawling the web so that user is benefited by updated and latest information. Due to the competitive nature of the search engine business, the designs of these crawlers have not been publicly described. But, there are two notable exceptions to this: the Google crawler and the Internet Archive crawler [20].

Crawler Architecture

The crawler architecture shown in Figure 2.3 has the following components with their respective functions as follows [11, 20] :

1. A URL frontier module is used to store the current URL fetched from current crawl, this URL is placed in the queue for next crawl activity.
2. A DNS resolution module to resolve URL address of a websites (*i.e.*, a resource located at a given server web, if an already requested URL will be fetched from cache.)
3. A fetch module that uses the HTTP protocol to retrieve the web page at a current URL, this module downloads the web page into a disk.
4. A parsing module is used to parse the HTML text out the fetched / downloaded web page.
5. A duplicate elimination module that determines whether an extracted link is already in the URL frontier or has recently been fetched (*i.e.*, make sure link has a visited or other status).
6. Robot template is used to govern the link follow rules.
7. A URL filter component is responsible to detect and extract links out with improper formats, otherwise malformed URLs can be encountered in the fetch module and will interrupt the crawling process (Exception handling).

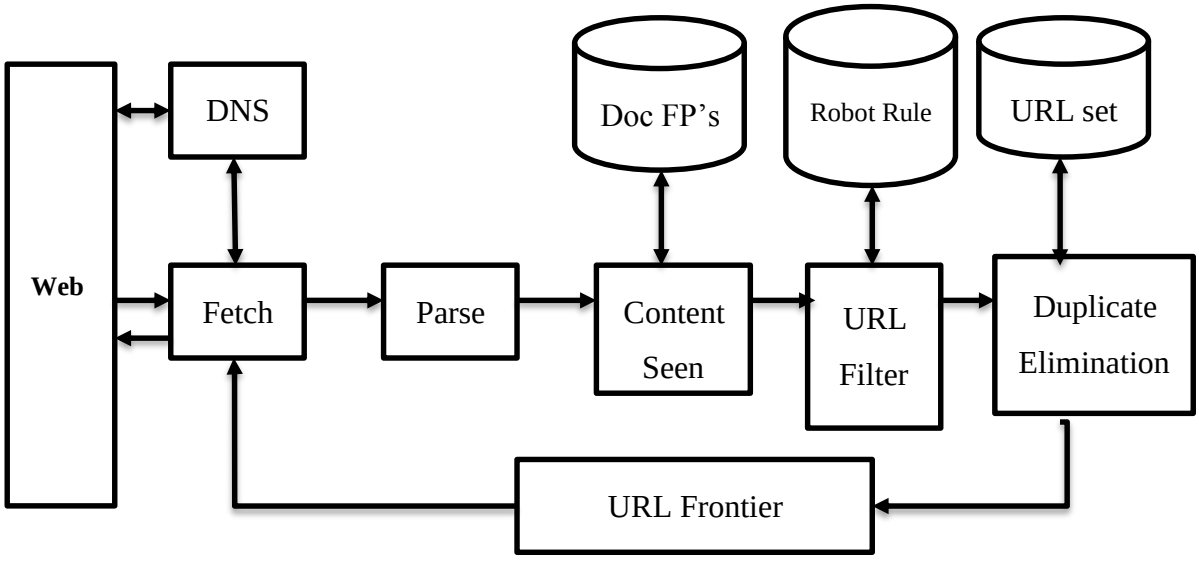


Figure 2.3: *Crawler Architecture*

Challenges for Web Crawler

Deploying and running a web crawler is a challenging task. There are tricky performance and reliability issues and even more importantly, there are social issues. Crawling is the

most fragile and expensive operation since it involves interacting with hundreds of thousands of web servers and various name servers which are all beyond the control of the system [21]. A simple web crawler process is simple; given the fact that following links and downloading contents as basic building blocks. Despite this fact of simplicity in the process, web crawling has many inherent challenges [22].

Scalability; Crawlers that seek broad coverage and good freshness must achieve extremely high throughput often achieved using different distributed machines. **Content Selection tradeoffs;** the crawler must balance competing objectives such as coverage and freshness, while obeying constraints such as per-site rate limitations. A balance must also be struck between exploration of potentially useful content, and exploitation of content already known to be useful. **Social obligations;** Crawler should not impose too much of a burden on the web sites they crawl. In fact, without the right safety mechanisms a high-throughput crawler can inadvertently carry out a denial-of service attack. **Spider traps;** The Web contains servers that create spider traps [11], which are generators of web pages that mislead crawlers into getting stuck fetching an infinite number of pages in a particular domain. Crawlers must be designed to be resilient to such traps.

Web Crawler Algorithms

Search engine algorithms are unique to every search engine. Basically, a search engine algorithm is a set of rules, or a unique formula, that the search engine uses to determine the significance of a web page, and each search engine has its own set of rules. These rules determine whether a web page is real or just spam, whether it has any significant data that people would be interested in, and many other features to rank and list results for every search query that is begun, to make an organized and informational search engine results page.

Breadth First Search Algorithm

This algorithm starts at the root URL and searches all the neighbor URLs at the same level. If the goal is reached, then it reports success and the search terminates. If it is not, search proceeds down to the next level sweeping the search across the neighbor URL at that level and so on until the goal is reached. When all the URLs are searched, but the objective is not met then it is reported as failure. Breadth first is well suited for situations where the

objective is found on the shallower parts in a deeper tree. It will not perform so well when the branches are so many [20, 23].

Depth First Search Algorithm

This is a powerful search algorithm which starts at the root URL and traverse deeper through the child URL. If there are more than one child, then priority is given to the left most child and traverse deep until no more child is available. It is backtracked to the next unvisited node and then continues in a similar manner. This algorithm makes sure that all the edges are visited once breadth. It is well suited for search problems, but when the branches are large then this algorithm might end up in an infinite loop [20, 24].

Web Crawling Techniques

Web crawlers are programs which traverse through the web searching for the relevant information [19] using algorithms that narrow down the search by finding out the most closer and relevant information. By applying web crawling techniques a specific set of web content can be analyzed, or crawler focus on selected topics of contents. The crawling techniques alone determines the type of information that needs to found.

Selective Crawling

The web is growing every day and every bit of a second. Sadly, research [25] has been showing that no search engine was able to cover the entire web. Crawling the entire web content is a very challenge task, because fetching and indexing a larger set of web content can significantly have an implications on the scalability of the overall system and, consequently, on the cost of the required hardware and maintenance services. Thus, selective crawling focused on optimizing the available resources, by recognizing the relevance or the importance of sites or pages, and limiting fetching to the most important subset of pages that can be downloaded in a given amount of time [26].

Focused Crawling

A focused crawler is a refined selective crawler that searches for information related to certain topics rather than being driven by generic quality measures [26]. Focused crawling is also general purpose Web crawler that gathers as many pages as it can from a particular set of URL's. The goal of the focused crawler is to selectively seek out pages that are relevant to a pre-defined set of topics. The topics are specified not using keywords, but using exemplary documents. Rather than collecting and indexing all accessible web

documents to be able to answer all possible ad-hoc queries, a focused crawler analyzes its crawl boundary to find the links that are likely to be most relevant for the crawl, and avoids irrelevant regions of the web. Thus, reducing the amount of network traffic and download needed [20, 26].

2.3.3 Data Miners

The data miner component of a search engine often encompasses the following components; Indexer, Page rank and Query engine / processor. Analogy of data miners can be visualized as a metaphor in separation of Gold from other impurities. The Gold is the actual and relevance information to be gained, but the impurities are unstructured and rough crawled data.

Indexer

A search engine system indexes the document information, pulling out specific keywords to categorize it. The index is built from the information stored with the data and the method by which the information is indexed. Data about web pages are stored in an index database for use in later queries. The purpose of an index is to allow information to be found as quickly as possible. Some search engines, such as Google, store all or part of the source page (referred to as a cache) as well as information about the web pages, whereas others, such as AltaVista, store every word of every page they find [27].

Page Rank

The citation (link) graph of the web is an important resource that has largely gone unused in existing web search engines. PageRank is an excellent way to prioritize the results of web keyword searches. For most popular subjects, a simple text matching search that is restricted to web page titles performs admirably when PageRank prioritizes the results. For the type of full text searches, PageRank also helps a great deal [21].

Query Engine

Query engine serves as a focal point between the end user and the search engine system. It allows users to feed queries into the system using UI handler and the engine examines its index and provides a listing of best-matching web pages according to its criteria, usually with a short summary containing the document's title and sometimes parts of the text. Most search engines support the use of the Boolean operators AND, OR and NOT to further specify the search query. Boolean operators are for literal searches that allow the user to

refine and extend the terms of the search. The engine looks for the words or phrases exactly as entered. The usefulness of a search engine depends on the relevance of the result set it gives back. While there may be millions of web pages that include a particular word or phrase, some pages may be more relevant, popular, or authoritative than others [27].

Distributed Crawling

A single crawling process even if multithreading is used will be insufficient for large scale engines that need to fetch large amounts of data rapidly. When a single centralized crawler is used all the fetched data passes through a single physical link. Distributing the crawling activity via multiple processes can help build a scalable, easily configurable and fault tolerant system. Though, distribution help in building scalable crawler and utilize hardware and software resource efficiently, it might have additional overheads. For example, when using separate machines across geographically different locations results in a significant overlap among the collections of fetched web documents [26].

Robot Protocol

Web sites also often have restricted areas that crawlers should not crawl. To address these concerns, many web sites adopted the Robot protocol, which establishes guidelines that crawlers should follow. Over time, the protocol has become the unwritten law of the Internet for Web crawlers. The Robot protocol specifies that web sites wishing to restrict certain areas or pages from crawling have a file called robots.txt placed at the root of the web site. The ethical crawlers will then skip the disallowed areas [24].

Fork/Join Parallelism

As multicore computers (computers with chip-multiprocessors) be-come mainstream, techniques for writing and executing parallel programs have become increasingly important. Fork/Join parallelism is among the simplest and most effective design techniques for obtaining good parallel performance. Fork/join algorithms are parallel versions of familiar divide and conquer algorithms. The primary goal of task scheduling in multiprocessor system is to minimize the total execution time, so that the process can achieve maximum speed-up and efficiency to carry out task. In fork/join parallelism approach a pool of worker threads is established. Each worker thread is a standard ("heavy") thread that processes tasks held in queues. Normally, a number of worker threads can be created based on the availability and capacity of the CPUs (cores) on a system. The

individual thread instances are lightweight executable class, not instances of threads to reduce memory requirement [28].

Work Stealing Scheduling

The heart of a fork/join framework lies in its lightweight scheduling mechanics. Work stealing has proven to be an effective method for scheduling fine-grained parallel programs on multicore computers [29]. For efficient execution of a dynamically growing multithreaded computation on a parallel computer, a scheduling algorithm must ensure that enough threads are active concurrently to keep the processors busy. Simultaneously, it should ensure that the number of concurrently active threads remains within reasonable limits so that memory requirements are not unduly large. Moreover, the scheduler should also try to maintain related threads, on the same processor, if possible, so that communication between them can be minimized [30].

Advantages of using Work Stealing scheduling:

- Dynamically balance the load across processes
- Maintain utilization even when competing with other programs for resources
- Handle even worst possible scheduling of processes

2.4 The Deep Web vs. Dark Web

The term "Deep Web" has been vaguely referred in description. But, generally it meant to refer content which is not reachable by standard link-based search engines like Google. Many studies have been conducted in crawling and extracting content form the Deep Web which has been characterized by a hidden content behind dynamic HTML forms [31, 32]. In broader sense, contents hidden behind HTML forms; normally made up of domain specific databases, dynamic content, unlinked content, private web, contextual web, limited access content, scripted content and non-HTML/text contents are factors to existence of to the Deep Web [31, 32, 33].

The Deep Web is also believed to be the biggest source of structured data on the web and hence accessing its contents has been a big challenge in the data management community [23]. According to Bright planet [24], Deep Web contains 400-500 times more information and 15% larger visit capacity than that of Surface Web. Moreover, the quality of data is also relatively higher. Another research indicated that search engines have reached only

0.03 percent of the information that is available; it's like a tip of the iceberg. Bergman argued that still much of the rest of the Web content reside in the Deep web [24]. There are many terminology used to describe the "Deep web", as the "Invisible web" and "Hidden web" and the relation with "Dark Net / web" will be discussed on following section.

The Dark web refers to any web page that has been concealed to hide in plain sight or reside within a separate, but public layer of the standard Internet [24]. Darknets are composed of host machines that cannot be accessed by conventional means, which is why the content hosted is typically not indexed by traditional search engines like Google and Bing. Virtual private networks are another aspect of the Dark Net that exists within the public Internet, which often requires additional software to access. The term "Dark Net" interchangeably used to refer "Dark web"; Websites found on Dark Net networks.

TOR is a popular 'dark net', a network that aims to conceal its users' identities and online activities. Hidden within the public web is an entire network of different content which can only be accessed by using the TOR network. I2P is also another most widely used Dark Net tool by anonym zing Peer-to-Peer networks on the Internet today. On TOR, web content and other types of services can anonymously be made available using hidden services. The network provides the possibility of publishing and maintaining anonymous server known as hidden services [34]. We will have a separate section that will discuss about hidden services below. Appendix A-D shows screen shot of DarkWeb websites.

I2P (Invisible Internet Project) is a message oriented, peer-to-peer based low latency anonymous communication network. The network was introduced to enable fully anonymous communication between two agents using the I2P network protocol. I2P was first introduced in 2003, having its origin in the Invisible Internet Project. A variety of applications inside I2P are available, e.g. anonymous web-hosting, web browsing, file sharing, email and many more. Using external services that are not hosted within I2P network requires the use of an out-proxy configured to work and understand the I2P protocol [35].

2.5 TOR Network

One of the main challenges facing users on the Internet today is maintaining their privacy, particularly in the face of online investigation. The Internet has revolutionized how we

communicate and achieve our tasks on our day to day activities. It has also opened a nature that can attract many entities that have preying eyes on users' personal information for commercial and other uses. It's known that online users are highly vulnerable to a variety of online threats of different kind. Anonymity networks have emerged as a solution to allow people to conceal their identities online. This is done by providing unlink ability between a user's IP address, his/her digital fingerprint, and his/her online activities. Allowing the user to be anonymous means that the user is unidentifiable within a set of users.

Onion Routing Protocol

The goal of anonymous communication is to solve the traffic analysis problem. Traffic analysis can be defined as follows [34]:

"*The problem of keeping confidential who converses with whom, and when they converse."*

Two of the most widely used anonymizing networks, TOR [16] and I2P [17] are widely using onion routing to achieve anonymity for low-latency applications. The Onion routing is one of the established techniques to provide sender- or receiver-anonymity [3]. It is especially applicable for building low-latency anonymous communication systems for example, web browsing.

The Onion Router (TOR)

TOR [4] also known as Third-generation Onion Router was invented on the mid-1990s by U.S. Naval Research Laboratory with the purpose of protecting U.S. intelligence communications online. The TOR, called The Onion Routing project, or TOR project, was launched on 20 September 2002 [4]. TOR is the most widely used privacy-preserving network, serving millions of people for more than ten years. It is a type of low-latency anonymity network which is based on the concept of onion-routing [3] design, where traffic is forwarded through a random circuit of routers and multiply encrypted through hops, with each router or hop removing one layer of the encryption at a time.

The communication establishes a path through the network which is the tunnel that is constructed in a telescoping fashion, so that each router knows only the previous and the next router in the path. In simple term the first at the entry router knows the source of the tunnel, but not its destination. *i.e.*, where the next route is destined, and the last which is

exit router knows the destination but not the source. The anonymity of a TOR circuit is compromised if the adversary can watch the two ends, the entry and exit, of the circuit as shown in Figure 2.4, the design of the TOR network with the onion routing anonymity model. The TOR network; Clients running Onion Proxies contact the directory servers periodically to get a list of onion routers and their descriptors. Clients then use TOR's router selection algorithm to build circuits. A circuit consists of three hops: entry guard, middle and exit. The exit performs the TCP connection on behalf of the user to the user's destination. TOR network today consists of approximately 6000 volunteer-operated routers [36], known as Onion Routers (ORs). TOR clients randomly select three of the roughly 6000 relays comprising the current TOR network, and create a cryptographic circuit through these to connect to Internet services. Since only the first relay in the circuit sees the IP address of the client and only the last (exit) relay sees the IP address of the destination, this technique separates identification from routing. To offer an onion service, web (or other) server creates TOR circuits to multiple Introduction Points that await connection attempts from clients. Figure 2.4 [36], shows the flow of communication in TOR network. A user wishing to connect to a particular onion service uses the onion address to look up these Introduction Points in a directory system. In a successful interaction, the client and onion site or hidden service then both create TOR circuits to a clients elected Rendezvous Point.

The Rendezvous Point mates their circuits together, and they can then interact as ordinary client and server of a web connection over this rendezvous circuit [37]. To use TOR all users need to install software called "TOR browser". TOR users are able to visit the WWW anonymously using this software. When accessing a WWW from the TOR network, TOR help user to hide his identify by making the connection masked through of the selected TOR's exit nodes. Often Journalists, Activists and Whistle-blowers, victims needing online support groups, and other users who are simply do a private information search use TOR. Using TOR software to access the Internet comes with drawbacks. These drawback is when a TOR based Internet connection used it will consumes more bandwidth than usual (*i.e.*, the speed of the network may be slower).

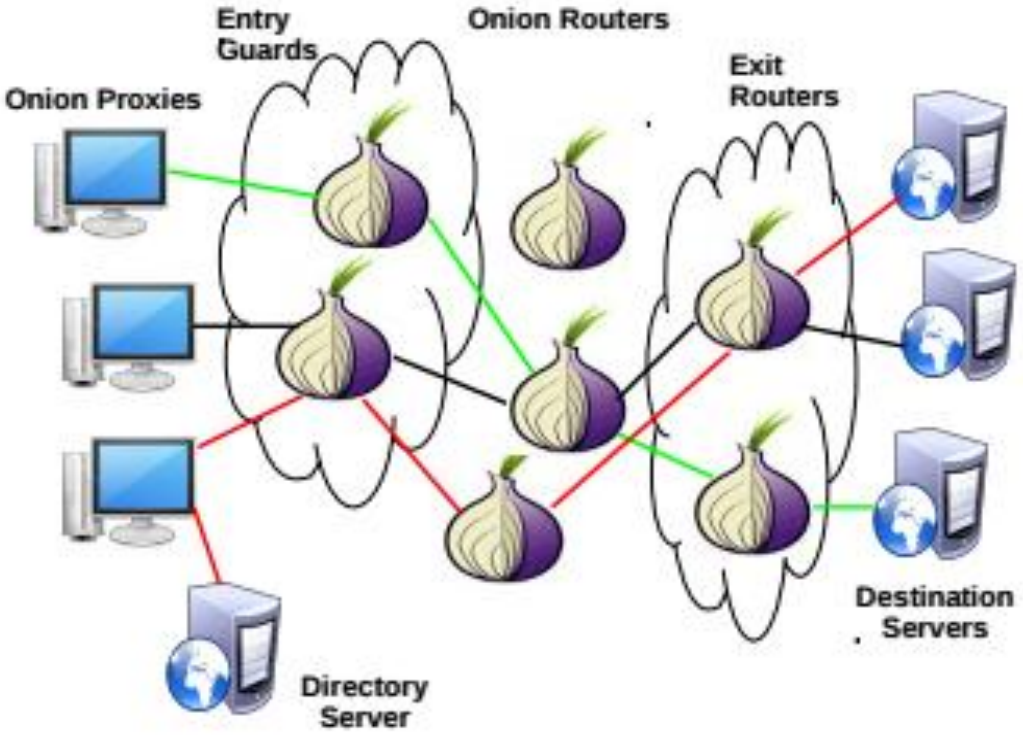


Figure 2.4: *Flow of communication on TOR network*

TOR Hidden Services vs. DarkWeb

The term "Dark web" most often is used to refer the TOR network, the fact that TOR network provides similar web surfing services like the Clearnet by using hidden services. These hidden services provide Web, FTP, SMTP, and chat services and they are not accessible without TOR software and hidden. According to a research which indicated that the term "Dark corner of the Internet", "Dark web" and "Dark Net could be used inter-changeably to refer to Deep Web [25]. TOR is an example of a system that exploits the absence of a non-delegated namespace within the global DNS system for its internal use. Hidden services, a unique feature within TOR, provide additional anonymity for users to communicate with servers. To identify these services, TOR uses the .onion name space [38]. TOR is an example of a system that exploits the absence of non-delegated namespace within the global DNS system for its internal use. Hidden services, a unique feature within the TOR, provide additional anonymous for users to communicate with servers. To identify these services, a user has to use .onion namespace [38].

TOR is capable of providing anonymity to servers, which are configured to receive inbound connections only through TOR; these are generally referred to collectively as TOR Hidden Services in research literatures and as the Dark web / Onion Sites in the popular press and on this document. TOR's hidden services let users publish web sites and other services without needing to reveal the location of the site. This is equivalent of the surface web resource (*i.e.*, service like web pages accessed without the domain suffix .onion). Since the onion site only communicates over TOR circuits it creates, this protocol makes anonymous commination (hides actual location) as 'hidden service'. Such services are services that separate their reachability from the identification of their IP (Internet Protocol) addresses. Figure 2.5 [39] shows how the Dark Net and Hidden Web found on Internet. Hidden service is a server providing an HTTP, FTP or SMTP services deployed on the TOR network. When publishing hidden service, first the server need to generate a public/private key pair and must choose some routers randomly as introduction points. Assume a user wants to advertise its services residing on TOR network, need a digitally signed descriptor containing the introduction point information and its own public key as public URL to access the advertised hidden service (website, ftp, email, chat etc.). Based on the contents of the descriptor and a validity time period t, a descriptor ID will be generated, and then the descriptor will be published in a Distributed Hash Table (DHT) hash ring that is formed by hidden service directories.

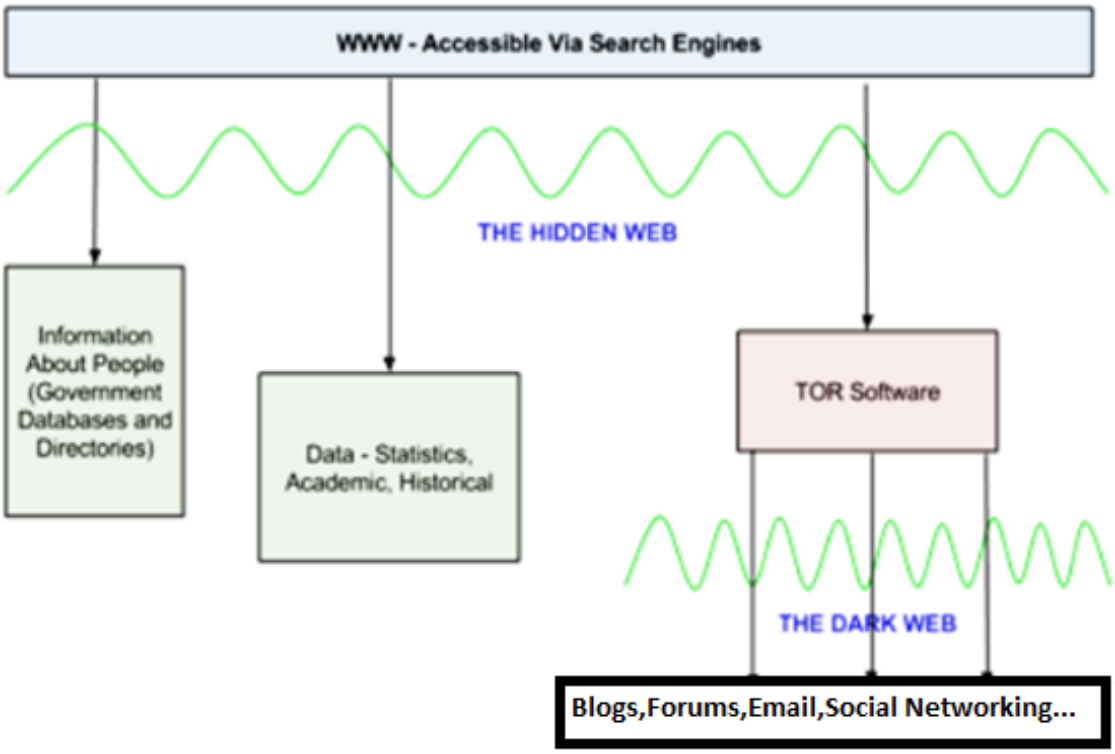


Figure 2.5: *The Deepest and Darkest Corner of the Internet*

The hidden service computes the directory responsible for holding its descriptors based on a closeness metric between the descriptor ID and the directory's fingerprint, which is the SHA-1 hash of the directory's public key [34]. When route requests to hidden services, a namespace is used to identify the resolution requests. A namespace under a non-delegated (pseudo) top-level-domain (TLD) of .onion was elected. TOR network is also designed to prevent .onion requests from leaking into the global DNS resolution. But, there are other important features to the .onion system [37], notably self-authentication and the advantage of fast, cheap, flexible and secure when compared to alternatives such as the standard use of TLS (Transport Layer Service) with certificates.

The hidden service publishes its onion address, which is an address of the form *32rfcibqhs6glimg.onion* where *32rfcibqhs6glimg* is a truncated hash of the hidden service public key that is generated automatically when deployed [34]. Unlike conventional web URLs, onion addresses are inextricably connected to the site authentication key. This means that if one has publicized the onion address, e.g., through blogs, Twitter, or Facebook, people following those address links will not be vulnerable to hijack due to key authentication [37]. Crawling the full collection of onion addresses is prevented by several objective reasons: firstly, many hidden service operators want to remain hidden. As a result, they rarely link to each– which my hold back traditional crawling to found them easily [40].

Improving TOR

Despite its current great potential, TOR has long-term sustainability problems. As more people become more privacy aware, a future TOR network should have the capacity to serve a significantly larger number of users. Because of several aspects of TOR's design, users currently experience inconvenient performance that manifests itself in the form of large and highly variable delays and download times experienced during web surfing activities. Such delays can be discouraging for users who wish to use it on a daily basis. The challenge to improve the TOR experience has been taken up by the research community, who has proposed dozens of research proposals and enhancements. According to a survey by Alsabah and Gold Berg [34] provided a survey of many of the performance and security issues as well as suggested improvements for TOR. The study reviewed the general network structure and footprint of the TOR network. Explained a brief technical overview of How TOR works? And investigated on the intended privacy and threat model

of TOR. More interestingly the study covers each component that are glued to make TOR network and indicated a options to make a lot of a room for improvement on performance and quality by reviewing different researchers work and indicating an intended solutions accordingly to the research done on a particular area of weak component.

2.6 Data Analysis on TOR

Data Analysis can take two parts. The choice can be using a Traffic analysis or actual content. Traffic analysis focuses on network infrastructure that make up the entire network and the software components (*i.e*., packets) involved. But content analysis uses actual data that is established on a given network using techniques like data mining, information retrieval, text and topic classifications. Web crawling and search engine technology helps for data acquisition and collection in content analysis research.

Content Based Analysis

Content analysis has its own approach to analyzing data that stems largely from how the objection of analysis, content, is conceived. According to Krippendorff [41], content analysis is defined as follow:

"Content analysis is a research technique for making replicable and valid inference from text (or other meaningful matter) to the contexts of their use."

The process often involves a set of technical procedures to carry out the task of analysis. Also, the results of a content analysis are expected to be reliable as expected to increase the researcher's insight on particular phenomena *e.g.,* TOR.

Traffic Based Analysis

TOR is a popular low-latency anonymous communication system. However, it is currently abused in various ways. TOR has been growing and consists of around 3800 volunteer TOR routers as of July 2013 [42]. It serves hundreds of thousands of users and carries terabyte of traffic daily.

Traffic related analysis on TOR is a most well researched area. A research by Ling *et al.* [42], aimed to design and implement a novel system, TorWard to provide an insight to discovery of malicious traffic over TOR. According to Ling *et al.* [42], TorWard: a malware detection system for TOR that can avoid legal and administrative complaints and allows the investigation to be performed in a sensitive environment. An IDS (Intrusion

Detection System) is used to discover and classify malicious traffic. It consists of a NAT (Network Address Translation) gateway and a TOR exit router behind the gateway. An IDS (Intrusion Detection System) is installed on the NAT (Network Address Translation) gateway to analyze the exit traffic before it is rerouted into TOR. This mean more malicious content can be found on TOR network among the malicious traffic discovered it includes P2P (Peer-to-Peer) traffic, malware traffic (e.g., botnet traffic), DoS (Denial-of-Service) attack traffic, spam, and others.

2.7 Amharic Language

Amharic (አማርኛ), the official language of Ethiopia, is a Semitic language that has the greatest number of speakers next to Arabic [43]. The Amharic alphabet is called Fidel, which grew out of the Ge'ez abugida-called in Ethiopian Semitic language. In modern written Amharic, each syllable pattern comes in seven different forms (called *orders*), reflecting the seven vowel sounds [44]. In spite of the relatively large number of speakers, Amharic is still a language for which very few computational linguistic resources have been developed. Written Amharic uses a unique non-Latin based syllabic script called "Fidel" or "Abugida" which has originated from the Ge'ez alphabet (the liturgical language of the Ethiopian Orthodox Church) [45]. Written Ge'ez can be traced back to at least the 4th century A.D. In the modern Ethiopic script each syllable pattern comes in seven different forms (called orders), reflecting the seven vowel sounds. The first order is the basic form; the other orders are derived from it by more or less regular modifications indicating the different vowels. There are 33 basic forms, giving 7*33 syllable patterns (syllographs), or fidels [46].

Word formation involves pre-fixation, suffixation, in-fixation, and reduplication among others. A significant large part of the vocabulary consists of verbs, and like many other Semitic languages, Amharic has a rich verbal morphology based on tri-consonontal roots with vowel variants describing modifications to, or supplementary detail and variants of the root form [47]. Subject, gender, number, etc are also indicated as bound morphemes on the verb, as well as objects and possession markers, mood and tense, beneficative, malfactive, transitive, dative, negative, etc. Amharic nouns (and adjectives) can be inflected for gender, number, definiteness and case, although gender is usually neutral [47].

2.8 Language Identification of Web Pages Based on Improved N-gram Algorithm

With the explosion of multi-lingual data on the Internet, the need and demand for an effective automated language identifier for web pages is demanding. Language identification is necessary when processing linguistic features from a content. For example, machine translation systems uses language identification to detect the source language correctly before the source text can be translated to another language [44].

There are three different approaches to generate language models and five different methods for language classification. The first approach generates language model based on "short words". It uses only words up to a specific length to construct the language model. The idea behind this approach is that language specific common words having mostly only marginal length. This approach uses tokenization and extracted all words with a length up to five characters that occurred at least three times from one million characters of text for European languages. In this approach a shorter words four or fewer characters, for thirteen Western European languages were considered. The second approach generates language model is based on "frequent words". It uses a specified number of the most frequent words occurring in a text to construct the language model. For instance, the most frequent one hundred words were used, also the most frequent one thousand words were used. The third approach generates a language model based on "n-gram". An n-gram is a subsequence of N items from a given sequence. The generated language model is used as the input for language classification method [48].

The general paradigm of language identification can be divided into two stages. First, a set of language model is generated from a training corpus during the training phase. Second, the system constructs a language model from the target document and compares it to all trained language models, in order to identify the language of the target document during the identification phase. Figure 2.9 shows a flowchart for language identification process.

2.9 Summary

Most research works targeted on TOR were mostly focused on Traffic analysis. Traffic based analysis research on TOR network aimed in protecting, breaking and predicating further anonymity problems and providing solutions. There are a few research to date that focuses on content analysis of the TOR network than traffic analysis. In TOR web contents are organized by means of hidden services. The fact that TOR allows its users to offer various Internet services like web publishing or messaging while keeping the location hidden using hidden services , a number of websites , forums and services exist on the network. A work [48] also argue that using n-gram based language identification at byte level would yield a better result and accuracy in detection of language on different web pages.

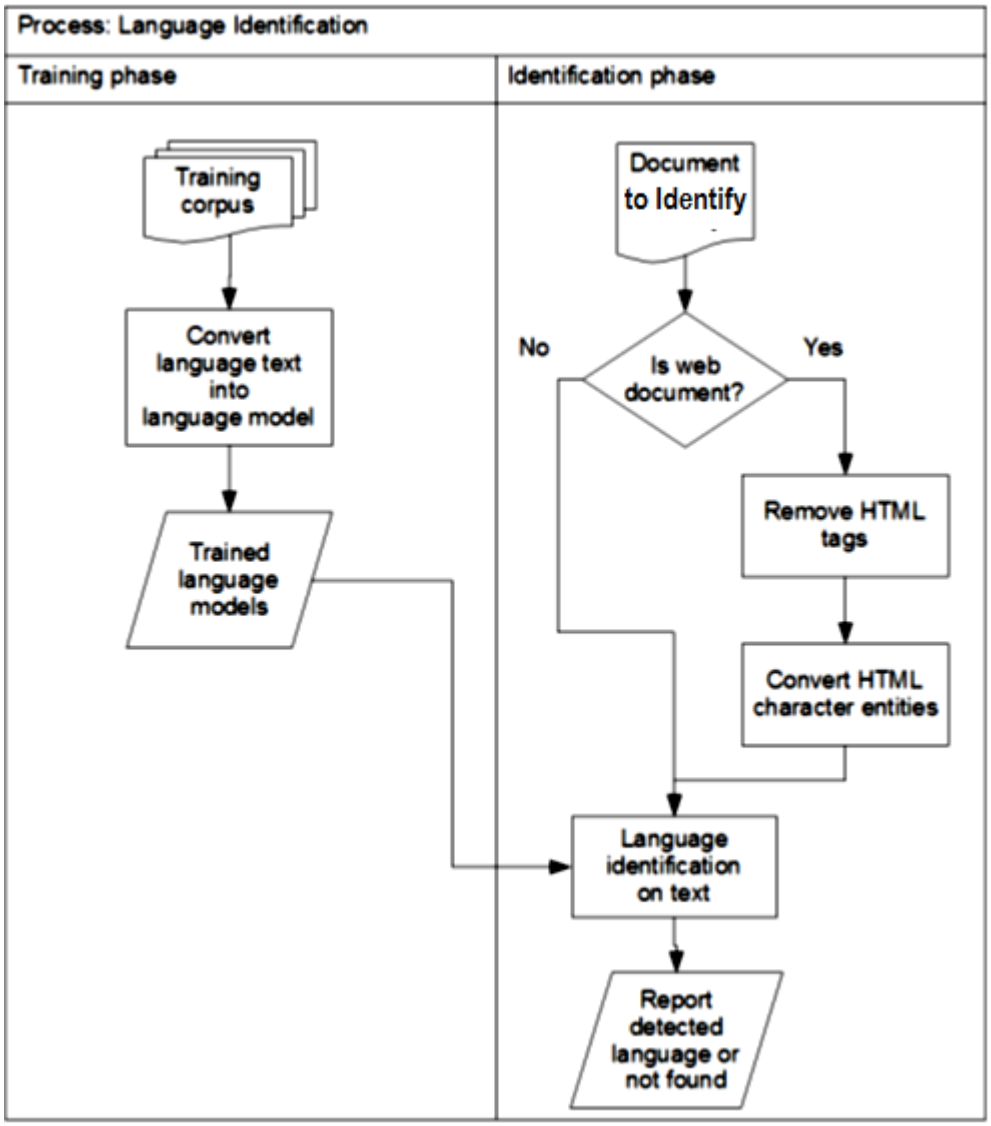


Figure 2.6: *Flowchart for Language Identification Process*

Chapter Three: Related Work

3.1 Introduction

In the chapter, we will review researches related to crawling of Dark Web contents on TOR network. TOR has long since been attracting researchers mainly due to its unique (anonymous) nature. A crawler come in handy when an automated content analysis is like for web documents is needed. It provides the analysis process a chance to collect and examine larger dataset. Analysis is also best determined on the collected dataset using automated software agents like crawlers. The crawling process in other hand requires details for design like technique of crawling, the part of the content to extract (e.g., topic and text classification is required), selection of which content to be downloaded, and the type of the network and security issues to be dealt with. The need for IR system which will do automatic content discovery and enables user to query over an organized data by monitoring and applying linguistic features, is a gap that need to be filled by a TOR based IR system. Section 3.2 automatic content analysis studies will be reviewed each of them with their pros and cons, section 3.3 summarizes the reviewed studies and outlines the gaps to be filled in our study.

3.2 Automatic Content Analysis of TOR Network

Recent studies on TOR were only focused on how TOR network is being used were by monitoring network traffic [6, 49, 50] than published contents on different DarkWeb sites. Some are more of or perform a content analysis by using crawlers of a single dark web market place. Other are more from determining a financial and economic outcomes [51, 52] than addressing the problem of surfacing its content to end users. Research [53] indicates that there is still a need to systematic way of monitoring, discovering contents of the network, either by using automated solution like web crawlers or other means.

A research by Biryukov *et al.* [10], applied content analysis to determine the landscape of TOR hidden services or Dark web. The research examined, 39824 hidden service descriptors using a known flaw on the network and an exploit on the protocol. A port scanning exploit and flaw were used to gain information on hidden services to look for open ports in the HTTP services. This research used crawling but has not given the methods of crawling or the architect of a crawler model. Despite this lacking fact, these research identified Dark Websites in 17 different languages and classified the content based on topic

presented and given a percentage of the general content figure based on content per topic distribution. The research lacks to have a description of the technique used to acquire content when a port scanning technique should not be the case in acquiring the actual web content. Another cons of these work it does not explicitly describe how the examined Dark Web site's content data were organized. The work is more of exploiting the TOR infrastructure to find hosted website by identifying on known port like 80. A notable work of this research is it has identified TOR web site that can be easily de-anonymized using misconfigured SSL certificate information, an open port that is often related to 'Skynet' botnet which has been found during the scan.

A work by Christin [51], a research conducted with aim of crawling and content monitoring analysis of underground marketplace for financial and economic projection of the selected ecommerce (Silk Road was a website found on dark web that facilitate anonymous ecommerce transactions). Christin applied crawling technique prior before analysis the content periodically form the date February 3, 2012, and until July 24, 2012. The crawling was to performed daily using an incremental mode, that is, ignoring pages that had not changed from one crawl to the next, each of these crawls ran, on average, for about 14 hours. The daily crawl took slightly over 3 hours considers as fast and the slowest took almost 30 hours, which resulted the other day crawling to be canceled to avoid overhead on the web site. The crawl process completed in about 48 hours and corresponded to approximately 244 MB of data, including 124 MB of images. The drawback of the work is it only crawled the individual pages like "item," "user" (*i.e.*, seller) and "category" form a single dark website. The challenge on this work was using the same open onion route network circuit to access the Silk Road website, often times during the crawl process (*i.e.*, traffic can easily be detected). However, the work addressed this potential issue by ensuring that all circuits, including active circuits, are periodically discarded and new circuits are built and they make an adjustment on the crawl starting time by taking some random time between 10pm and 1am UTC. The work analyzed over 24,000 items, and parsed over 180,000 feedback messages. Further , the study discovered that the number of active sellers and sales volume are increasing, corresponding, when averaged using custom measurement interval to slightly over USD 1.2 million/month for the entire marketplace, which in turn represents around USD 92,000/month in commissions for the Silk Road operators. The author argued indicated that content analysis of the market place website was using a crawler. But the crawling architecture and the detail of the crawler application

was not presented. It was not used for analysis other similar e-commerce dark web sites with similar purposes other than the selected dataset because, basically it lacks providing a crawler architectural that would help other works on the area.

A research conducted by Semenov [40] described an application of the generic social media monitoring system for gathering and analyzing data from TOR hidden services, maintained in the Finnish language. It was an exploratory research that reveals what can be crawled from TOR and what kind of contents there exists. The research tried to describe hidden services and dark website based on content as well as indicated a possibility of tracking users of TOR's hidden service. It has crawled only two social media dark website or hidden services: the imageboard "Thorlauta" and the forum "Suojeluskunta". The draw back on these work is it lack discuss crawling technique, How data is collected and organized when the crawler is used. Despite this, the work provided a linguistic analyzer model and developed a software system for logistic analysis using a three-layer model, where immediate social reality is modeled by the first level – social media sites, which are modeled by multi-relational graph, which is modeled by a third-level model, a database implementation of the graph.

Typically Semenov [40] and Christin [51] , described crawlers deployed within the TOR network itself rather than trying to crawl outside the TOR network. They focused on specific and limited number their targeted dark websites when crawling and analyzing their contents. They have not described the crawler design for dark web and the challenge to do so when crawling dark net like TOR. This includes lack of algorithms used or have to be used. This is the gap that we will try to fill on this thesis.

Topic based social network analysis research by L'Huillier *et al.* [54], conducted a research for content analysis on limited number of selected dark website forums. The work tried to study extremist groups and their interaction by targeting forum dark websites. The goal was by analyzing social network forums it helps government and other stockholders for developing counter terrorism applications. It aimed at addressing the topic-based community key-members extraction problem. According to the research the LDA (Latent Dirichlet Allocation) algorithm is proposed to discover different topics from Dark Web forum sites, but not further social network analysis applications with these findings. The LDA approach was used after the completion of crawling the selected datasets. Even though the work has no architectural model used for crawling, gives an experimental result

for text and topic classification which shows a 14 months (Dec. 2008 - Jan. 2010) for a data that was used from the selected dark website forums. A major drawback of this work is, the crawling technique used, the data collection mechanism were not discussed, as well as no model of crawler were presented. As a good side of the work was it proposed a novel approach for topic modeling using LDA and argued that with as promising.

In more similar approach to L'Huillier *et al*. [54], research by Jennifer Xu *et al.* [55], conducted on topological analysis of terrorist group dark websites. In this paper, they proposed to apply network topological analysis methods on a collected the terrorist website data and to study the structural characteristics at the Web page level. Though, their data collection mechanism was not stated. The work [55] claimed forming a dataset of data on three collections of Middle-Eastern, US domestic and Latin-American terrorist websites. According to the work a crawling technique was used prior content analysis of the elected dark websites believed to be representing extremists. The work [55] provided an outcome on topological analysis based on structural relationship among the three collections of data set (*i.e.,* country wise). The major advantage of this work is trying to study terrorist groups on the dark net with content analysis have a potential to help governments develop counter measures. The drawback to this work is they have not given an architectural mode for crawler, especially the type of their analysis requires additional parameters to classify a given websites as extremist or not. Furthermore, the work presented a few dataset, due to lack of crawling technique. Crawling for additional Dark Web site would have a better chance of discovering similar contents for their experiments. The work used only the structural content found on the webpages (*i.e*., HTML tags) but not actual content or meanings of the words from the texts. Analysis of actual text content would have been useful in revealing additional information, than the structural formation.

A. Ríos *et al.* [56] , conducted a study on dark web portal overlapping community detection based on topic models. It argued that tools such as social networks analysis and text mining have contributed to the understanding of these kinds of groups in order to develop counter terrorism applications. A key application is the discovery of sub-communities of interests. The work believed using main topics from social networking websites content could be used to alert on a possible homeland security threat. However, most algorithms detect disjoint communities, which means that every community member belongs to a single community. Thus, final conclusions can be omitting valuable information which leads to wrong results interpretations. The work proposed a novel

approach to combine traditional network analysis methods for overlapping community detection with topic-model based text mining techniques. The novel method used for the advantages of LDA (Latent Dirichlet Allocation) to build improved and filtered networks is proposed. This approach considers a network filtering, followed by an algorithm to detect overlapping communities in networks by topic propagations. It is based the modification of that allowed to include semantics of the information gained. The experiments were performed using an English language based forum called "*Islamic Awakening*". The work shows that results obtained for finding overlapping communities using the above mentioned method is better than using traditional methods over a LDA-filtered network. The work argued their proposed methods were useful mainly when mining a related forums website (*i.e*., Jihadist), finding better communities, improving the understanding of the networks, and facilitating administration experience. The ideal contribution of this work was a community finding strategy which considers both structural properties of posted messages and their content semantics. Despite this fact, no particular crawler architecture were suggested and also the impact and challenge of linguistic features in analysis of the forum website were not presented. And, English language based dark website forum was only considered. The number of the dataset was only one.

3.3 Summary

Most of the related works were focused on single or limited number of website analysis specific forum, extremist and market place dark websites. Mainly the work lack in describing the clear and transparent approach of content gathering analysis, and organization than just mentioning a crawling technique was applied. Most of the review work share the main problem of not having a proposed design for their crawling model, while their means of data collection were indicated as crawling. And, most of this work have would yield a better result if limited number of dataset were not used. And, this problem could be solved by using a crawler designing a general or specific crawler for the problems discussed. Our work can help this gap to be filled by providing a crawler architecture for the Dark Web that is not limited to crawl and analysis specific and few number of websites. Biryukov *et al.* [10] ,tried to do a content analysis using port scanning to identify if hidden services for web contents are running on web server or not then collects web content for topic and text classification, but the work creates overhead to the analysis process as well as creates a security concern.

The Silk Road's study by Christin [51], and another by Semenov [40], shows the problem of limited dataset considerations. The entire TOR network was not crawled instead only single website types, which will harvest much better result in finding many other dark websites and hidden services deployed with e-commerce and illegal contents (*i.e.*, other similar website like the data set they took). Common to all research reviewed, the technique of crawling has not been explicitly described, since the aim can be clearly seen that it is for text and topic analysis only. Christin's work used a crawler to collect and gather analysis data and the crawler setup was by installing it inside a separate TOR exit node server (*i.e.*, they did not gather content from a Clearnet point of view). Overall, there is limitation on the crawling processes. In general, to the best of our knowledge, the researches done so far on TOR network does not cover about a designing of a Dark Web crawler and identification of Amharic contents from the dark web. The proposed system will be focused on proposing crawling algorithms, design components and provide an architecture for crawling process and for integration of offline language identifier to identify available Dark Web Amharic contents.

Therefore, we alleviate the problems by designing a Dark Web crawler. Our proposed system will not be limited to discover and analyze a few number of DarkWeb websites (*i.e.*, 1-4), unlike other related research works. In addition, there is no work done which provides a model for a search engine crawler specific for TOR's network other than reusing surface search engine crawler which basically have a problem in crawling onion based websites. But we consider a TOR content under Onion Routing Protocol. The proposed model enables wider portion of the TOR network content to be discovered, analyzed and downloaded. Finally, identification and collection of Amharic content from Dark Net websites is possible by integrating a language identifier on the proposed model of Dark Web crawler. Future works can integrate the model with search engine by adding only indexer and query engine components to make the proposed design as a complete search engine for Dark Web.

Chapter Four: The Proposed System

4.1 Introduction

As described in different research works, crawling task often involves different set of operations in collaboration by following systemic course of actions. At first, crawling seems a trivial task, but aiming an industry standard crawler requires a careful consideration of the network bandwidth, the protocol, the underlying computer resource, distribution, scalability and resilience (fault tolerance). In this Chapter, we propose an architecture for TOR search engine crawler.

4.2 Design Consideration

For designing a search engine crawler for TOR (Dark Web) we have the following assumptions. First, the nature of the networks as being low latent (*i.e.*, allows human-unnoticeable delays between an input being processed and the corresponding output providing real time characteristics) and the fact that it is isolated from the WWW; assuming a major drawback of speed and performance on the crawling process. The need for adding additional infrastructure like TOR client/server running to server network request were inevitable. Second, emphasis is given on how to resolve the Dark Net Web addresses (URLs) into their corresponding HTML pages. To address URL resolving issue we used a separate DNS cache client. Third, a TOR Controller component is used to detect and mitigate error that happen when crawling (*i.e.*, reset the TOR Client when connection timeout). Fourth consideration is we used a component to identify Amharic content after web documents are downloaded. Finally, the HTML page meta-tag and page title are extracted to for later search and information retrieval process.

4.3 The Proposed Architecture

In our design, we used different number of seed URLs to initialize the crawling process, then a URL queue is kept to track visited and non-visited links, once extracted out –and – in links from a seed URL, the next process puts all the extracted link into the URL queue and the thread manager will subsequently initialize instance of threads per URL until all links in URL queue are visited. The result from each of the crawler threads for individual webpages are recorder into a database. The task of downloading dark web page is carried out without waiting for the main crawler thread to finish. The downloader works by

fetching links from an already persisted data (database) to download and store it in file as compressed HTML format. Finally, an offline language identifier is used to identify the language from downloaded dark web HTML content. Based on the objective described in Chapter One our aim to design a crawler for Dark Web. Thus, we filter or discard Clearnet URLs by using a Dark Web address filter component. The architecture of the DarkWeb crawler with the integrated offline language identifier is depicted in Figure 4.1.

4.3.1 URL Queue

Theoretically speaking crawling a large portion of Dark Web seem an infinite and recursive task. In fact, crawlers do have a terminating condition and they halt when all URLs are visited. Therefore, a URL queue component is mandatory to keep track of the crawling process. In our design, the extraction of links from a given seed URL is going to be picked one by one unless the seed URL points to itself, which lead the crawler to halt soon. The mechanism is to keep all the extracted URL / links in memory. Thus, it helps not to keep crawler threads being trapped in infinite loop visiting a single URL over and over. To alleviate such problem we used a concurrent hash set of memory variable. Later, URL queue data is mapped to the database using concurrent persistence storage when it get visited as discussed in Section 4.3.5.

4.3.2 Fork Join Thread Manager

A Fork-Join Thread manager is the heart component of our design. This component allows crawling of web pages by implementing an algorithm called a divide-conquer to carry out individual crawling task in recursive manner. Not to confuse with Breadth First crawling technique but this component use the divide and conquer approach after getting a URL/s from a URL queue it recursively performs extraction of in-an out-links and the crawling continues recursively. Therefore, priority of links will be decided using Breadth First approach but recursive crawling is decided by the Fork-Join thread. The Fork-Join Thread manager component is responsible in managing, instantiating, reporting and assigning task to individual thread instance from a fixed thread pool size initialized prior staring the crawler. Each thread instance will be given a URL/s after a Link Extraction Module is called by thread instances of the Fork-Join crawler. The assigned thread instance process the URL and changes its status after processing and updates its visited status into the URL queue variable (*i.e.*, , shared among thread instances as thread safe). Later, the each URLs in URL queue will be persistence into the designed database storage when their visited

status changed. URL queue and the Persistence Storage (database) access will be allowed to all crawling thread instances thread safe.

In Fork Join thread instead of starting a new thread for every task to execute concurrently, the task can be passed to a thread pool. As soon as the pool has any idle threads the task is assigned to one of them and executed. Internally the tasks are inserted into a Blocking Queue which the threads in the pool are dequeuing from. When a new task is inserted into the queue one of the idle threads will dequeue it successfully and execute it. The rest of the idle threads in the pool will be blocked waiting to dequeue tasks. It monitoring each threads job status, assign new thread instance when there are available from the thread pool. When threads with less job are detected they will be assigned work stolen from other thread instances with greater tasks (*i.e.*, a crawler thread instance with greater number links). The work stealing algorithm is used to monitor individual thread of crawler in the Fork-Join Thread management approach. It is given that web crawling itself is recursive process and we gain the performance and efficient utilization of system resources and parallelism of tasks in muli-core processer using the fork join method.

The Fork Join thread manager implementation utilizes the Java's concurrency and Executors Service frameworks model to manage threads using a divide and conquer algorithm. The implementation of the fork-join framework provides a settings and options for determining the size of thread instances and other thread management operations. By using this implementation we have specified a fixed thread pool size with maximum of 15, 30, 64 and 100 thread pool size. The size is selected due to resource limitation on the current implementing machine. The architecture for the proposed Dark Web Crawler with Downloader and Offline Language identifier components is shown in Figure 4.1 and the algorithm for Fork-Join based Dark crawler is shown in Algorithm 4.1, respectively.

4.3.1 Link Extraction Module

The link extractor module is the newly introduced architectural component for crawling dark web in sense that it contains a DarkWeb Filer component. It is used to filter only e web address that belong to the dark web based on the onion suffix that every URL in dark web have on its parent domain name. After getting their domain URL extracted links are converted into their absolute and relative path.

MetaTag and Page Title Extractor

This component extracts the Meta information from a dark web website. It looks for a Meta tag in the HTML structure. It stores the extracted Meta information into a database by using Concurrent Persistent Storage manager component for the required association of the Meta information and the corresponding URL. In similar manner page title is also extracted and associated with URL in database.

```
Input: - seed URL
Output: - extracted URLs, Meta Tag, Page Title
Create list of Recursive Actions
IF (seedUrl IS Visited ) Then
  Open connection to the seedUrl
  IF connection timeout for 2 minutes Then
   Suspend all crawler threads for 10 second
   Send command using a Tor Controller  to Tor client
  End If
  PARSE HTML from seedUrl
  EXTRACT_META_DATA from seedUrl
  EXTRACT_PAGE_TITLES from seedUrl
  URLS = LINKEXTRACTOR extract all links from seedUrl
 FOR EACH URLs IN i
   linkTag = pickLinkAt ( i )
   IF  ( DarkWebAddressFilter to check for linkTag  ) Then
     IF ( linkTag NOT EMPTY AND linkTag NOT VISTED in UrlQueue ) Then
           Create child of fork-join crawlers based on ( linkTag , UrlQueue )
     End If
   End If
  End For
 Add seedUrl in UrlQueue as visited
 Use ConcurrentPersistenceStorageManager  to save (seedUrl, pageTitle , metatag)
 Invoke child of fork-join crawlers //repeat for all crawlers created
End If
```

Algorithm 4.1: *The Proposed Fork-Join Based Dark Web Crawler*

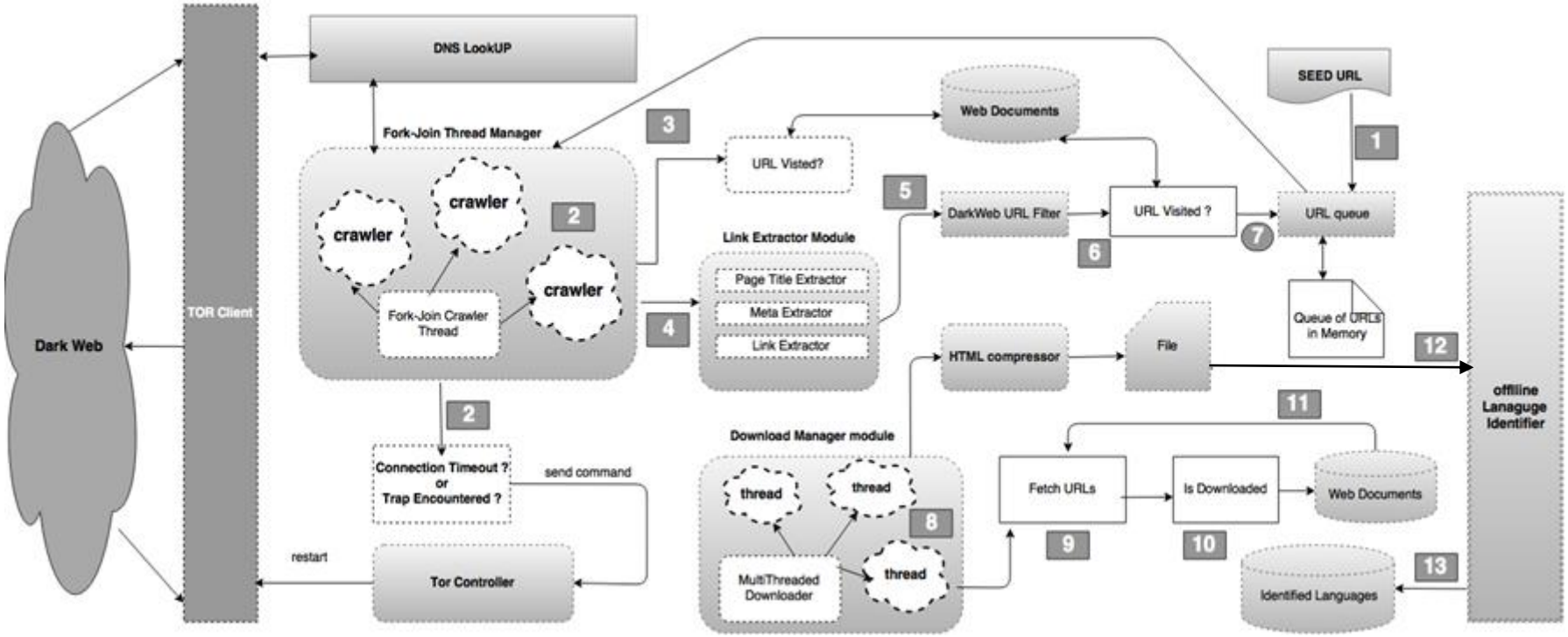


Figure 4.1: *Architecture of a Dark Web Crawler and Offline Language Identifier*

After extraction of links using the Link Extractor module we assign individual crawler to perform additional link extraction task on already discovered URLs / links. Assume, if we are not keeping track of URL/s or links when they get visited or not , then our Dark Web crawler will going to waste time and be trapped visiting the same URL over and over. This will decrease the performance, the quality of the crawler data as well as it make it no use for later search and retrieval. Thus, a Link Extraction module helps to avoid such problem.

The Link Extractor module interacts actively with the persistence storage media for URL/link addresses to check for if the crawler has been interrupted or stopped for some reason in the past and check for which URL was been the last in the URL Queue by checking for visited status of URL being hit using the Concurrent Persistence Manager component. After getting the hitting into the URL who is supposed to be in the URL queue from the past interruption it enables the crawler start its task from where it left off.

The Link Extraction module is implemented to look into an HTML structure from the crawled dark web documents using HTML tag parsing. The Link Extraction module helps by filtering URLs that doesn't belong to the DarkNet. It uses an HTML parser library to check on anchor elements (*i.e.*, "**a: href")** tags for each links and the content being hit. When a dark web content is loaded into memory it extract all links and their URL address. The Link Extractor algorithm is shown in Algorithm 4.2.

```
Input: - Url
Output: - extracted URLs
ALGORITM LINKTAGS LINK_EXTRACTOR (URL)
PARSE URL
URL_FILTER_RULE = [! EMPTY,! MALFORMEDFORMAT]
IF (URL_FILTER_RULE = TRUE)
  EXTRACT_ALL_LINKTAGS (URL) // return URL address in a link tag
  RETURN LINKTAGS;
ELSE
   RETRUN NULL;
End
```

Algorithm 4.2*: Link Extractor*

Dark Web Address Filter

When crawling .onion websites using the extracted links /URLs, one can encounter web address that are outside of the TOR network *e.g. www.blockchain.info*. But, we focused on our design to consider only extracted links that are pointing to TOR (onion) based host address. The rest are discarded. Thus, placing a link filtering mechanism for checking if a URL belongs to the DarkNet or the Clearnet is mandatory.

In our study we do this by checking the host address of each extracted link before it is added to URL queue in memory as well as in the database, so that we only have onion address kept. This will saves the time and resource (e.g., crawler threads) wasted in wandering into Clearnet and we meet our objectives by only crawling into the Dark Web.

```
1. Input: - URL
2. Output: - a status for the URL entered is an Onion or not
3. DarkWebAddressFilter ( URL )
4. HostName := getHostNameFromURL(URL); // gets the ULR's host domain name
5. If (Hostname ends with ".onion" ) then
6.  Return true; //its dark net
7. Else
8.  Return false;
9. End
```

Algorithm 4.3: *Dark Web Address Filter Component*

4.3.2 TOR Controller

Often when crawling the Dark Web crawler threads can be trapped in a loop because of a malfunction on the underlying TOR Client software or waiting for long for a requested URL connections to return the actual HTML content. Thus, to moderate such problems we used a TOR Controller component. The TOR controller sends command (*i.e.*, using network sockets) to restart, halt or change new circuit node forcing the underlying TOR Client to change its network setup. Furthermore, to moderate the crawling process we first suspend all crawler thread using the Fork-Join thread manager. Second, we wait for the result of the command. Finally, we resume all the suspended threads and will continue to crawl. To perform this procedure a separate control port is used to communicate with a TOR client.

4.3.3 Concurrent Persistence Storage Manager

Having URL/s extracted on main memory as hash set of current linked listed of URL queue would not be enough for the fact that keeping URLs in main memory (RAM) is volatile and overwhelming to keep all when number of URLs increase drastically. To alleviate this we used two main approaches. First, to keep all URLs into a memory variable of hash set using concurrent linked list, a hash set maintains a unique entry and would not allow URL/s to be repeatedly added. A hash set becomes thread safe when applying the concurrent linked list implementation (*i.e.,* a thread cannot access URL from the URL queue when another thread crawler is accessing), allowing reduced size of data to be accessed in linked list manner. Basically, threads gain access to the shared instance of the URL queue variable to be available for all crawler thread instances. And, keeping URLs in a hash set can reduce memory consumption and provide effective reference fetch from main memory. The linked list of hash set will help our crawler to keep track of FIFO (First in First Out) access of URL/s from the URL queue.

The second approach uses a persisted data storage of URLs. Even though, we still keep URLs in URL queue we face a problem of being volatile, and the main memory is expensive in terms of size compared to disk storage. Therefore, we used a persistence storage to keep track of URL/s visited and to keep additional information about each URL being visited. We used this component to record other attributes of Dark Web URL. This will increase the likely hood of our Dark Web crawler, to be used for search engine. For example a record parameter like page title can be used in a search engine when indexing.

The implementation of this component is using a *Static* type class (*i.e.,* no need to instantiate a class for each thread instance). By using the Basic Data Source Java library which helps when making database connection to the database server. Later, an already created connection object is pooled than creating a new connection ever time when a crawler or download thread requires it. The instance of the database connection object is shared across the system and accessible to all concurrent threads as thread safe controlled by the database server. The algorithm for the Concurrent Persistence Storage Manager is given in Algorithm 4.4.

```
Input: - isUrlVisted , urltoSave ,UrlMetaData,URL pagetile, customQuery ,
Output: - customQueryResults, FoundUrlResult , savedResult
Set currentConnection = null
Set urlConnectionPath = set database path from databaseUrl
If (currentConnection not  initalized) Then
    If (concurrent data source variable not initalized) Then
       Connect to database (databaseUrl)
      add connection to list of concurrent data source variable //for other thread access
    End if
 currentConnection = pool connection from the concurrent datasource variable
End if
If ( isUrlVisted not null) Then
 foundFlag = isUrl exists in database (isUrlVisted)
   return foundFlag
End If
If ( urlToSave not null) Then
 foundFlag = isUrl exists in database (urltoSave)
   If ( foundFlag is false) Then
    savedResult = Save ( urlToSave , UrlMetaData , UrlPageTitle )
    return savedResult
  End If
End If
If (customQueryResults not empty) Then // this time it is asked to check if URL visited or not
  customQueryResults = queryDatabase (customQueryResults)  // a download manager queries
  return customQueryResults
End if
End
```

Algorithm 4.4: *Concurrent Persistent Storage Manager*

The designed conceptual schema for the proposed crawler and offline language identifier is shown in Figure 5.3.

4.3.4 Download Manager

The Download Manager component is used to download and store dark web documents in files. The advantage is it enables further processing on available web files. For example, offline language identifications on web files can be done.

The design of the Download Manager uses multi-threading to download URL using individual thread instances assigned. It uses thread pooling approach to efficiently utilize the computer resource and increase the performance of download for identified onion sites. The download manager get links to download from the URL table from the deigned web database. It fetch links until all links in database are downloaded. We track this using *IsDownloaded* flag to and later it gets updated on the database when a link is downloaded. Therefore, there will not be a multiple download of a Dark Website.

The Download Manager can run from different machine/s allowing the downloading process to be distributed across computers. To achieve this a shared persistence data of URL/s shared among each instances of the threads is required. This approaches is less resource intensive, in relative to running on crawler and downloader using a single machine. The only requirement for our download manager component is the availably of visited URL data access, which could be satisfied by providing the URI of the persistence data media (web database).

When downloading web requests using individual links for contents happens. Each thread will open connection to its assigned URL for fetching content and pass it to HTML compressor for further processing (see section 4.3.5 below) and compressed HTML file is saved. The download threads are instantiated per URL basis, so that there will not be a multiple download of a Dark Net web page more than once. The Download Manager update the database by taking advantages of Persistent Data storage Manger‘s concurrent data access paradigm to avoid concurrency and parallelism issues. In our case we have limited to MAX number of downloader threads size to 10. The algorithm for a Downloader Manager is shown in Algorithm 4.5.

1. **Input:** - List of URLs, maximum Thread Size
2. **Output:** - Compressed Web Document
3. Algorithm Downloader Manager (listofurls ,maxThreadSize)
4. Initialize the Executer Service Thread manager with maxThread
5. **While** ListofUrls not empty **Do**
6. Fetch the web document from the url
7. Compress the web document
8. Save as compressed HTML file
9. Update the database
10. Point to next Url
11. **Loop**
12. **End**

Algorithm 4.5: *Algorithm for Download Manager*

4.3.5 HTML Compressor

A Dark Web site contains texts, multimedia, images and files data to represent it using an HTML format. When downloading a dark website contents its wise that we think which contains we interested in. Downloading all multimedia, files, texts and images into disk and store them for offline processing would requires a huge disk space if we are talking about millions of web page to be downloaded. This, this problem causes drawbacks to a search engine IR system, if they have limited number of disk storage. In addition, processing large data also requires intensive resource like CPU and memory for computing. Theoretically, our crawler is expected to cover all the web pages found on DarkWeb. Thus, in our proposed design we have added an HMTL compressor to reduce the size of dark web pages.

The HTML compressor works in collaboration with the Download Manager component. Reducing the disk size requirement will bring easy access of download web files for later retrieval and analysis. When downloading dark web's we used a text only scheme and downloading other contents like multimedia and image are not necessary to our study and are discarded.

4.3.6 Offline Language Identifier

The process of language identification of a web documents can be carried out using two methods. The first method is by using Meta-Tag and Charset information when parsing the HTML at crawl time. After that a search engine crawler may tend to do further analysis on the whole or portion of the parsed content by investigating its linguistic characteristic of Meta tag, HTML structure or portion of the content. This is referred as online identification. The second method is by using analysis of the document after downloaded and apart from the involvement of the crawler running. This method uses the web document's written text to analysis it linguistic characteristic by using N-gram method or other method. This is called offline identification.

In this research, the technique of checking the META tag and Charset information from HTML content is not used (*i.e.*, online identification). We used the G2LI as offline identifier. The name G2LI refers to a Global Information Infrastructure Laboratory Language Identifier [48]. The model of the language identifier used in G2LI is discoursed in Chapter Two, Section 2.9. It is a text language identifier that can be used to simplify on how to understand the type of language, script and encoding schemes a text content is written in.

In the design of offline language identifier we first modified G2LI to use a GUI. Second, we modified the architecture of the existing G2LI model by adding a database module for saving individual identification results into a database for all compressed HTML files. The modified design also uses our local data repository directory and available downloaded web documents are analyzed. The modified G2LI has been trained UDHR corpus for Amharic and other language support. Based on the training corpus the G2LI used a byte sequence of N-gram bytes that are generated and used to uniquely identify web documents later. The modified architecture of the proposed offline language identifier is shown in Figure 4.2 [48].

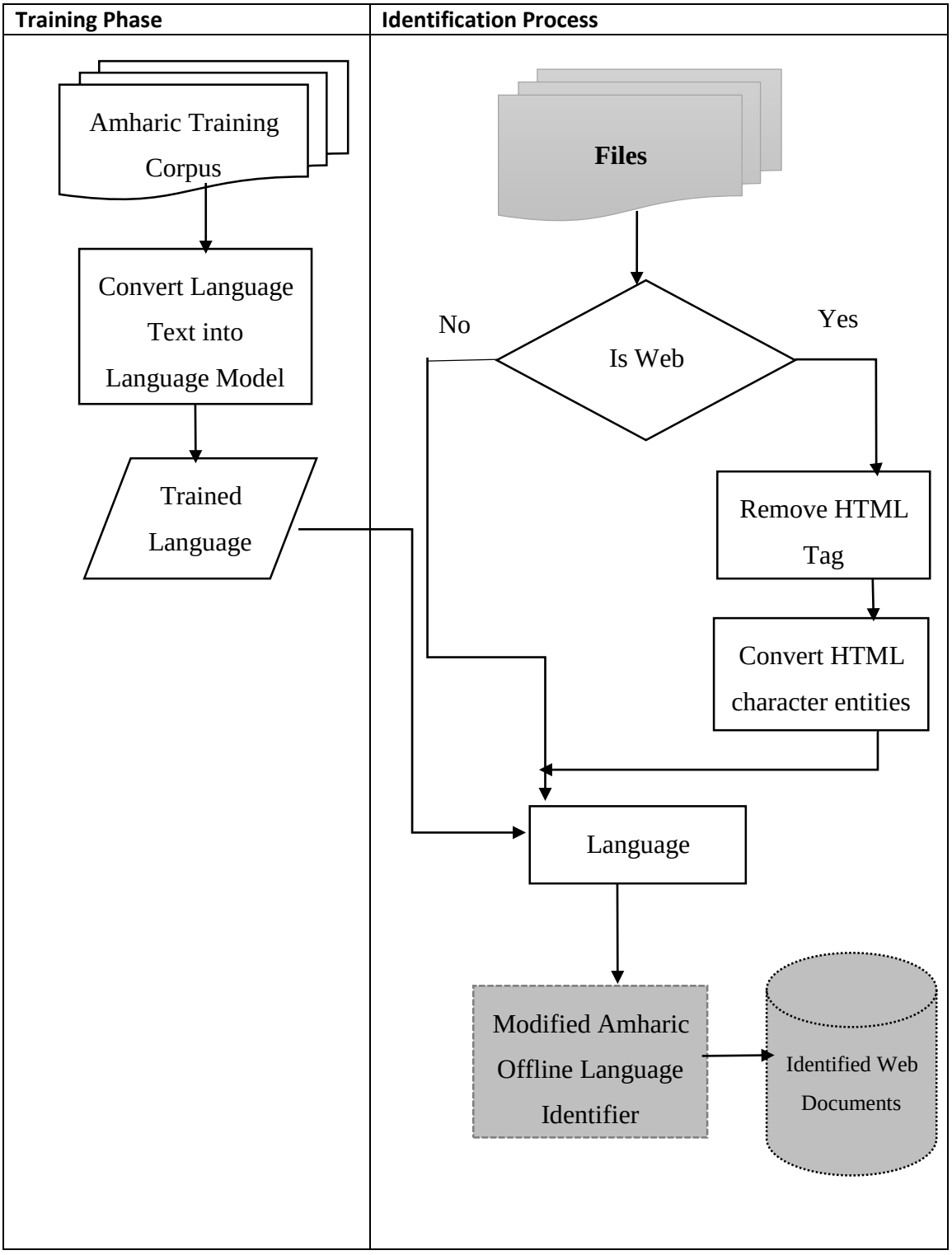


Figure 4.2: *The modified architecture of the Offline Language Identifier using G2LI*

4.3.7 TOR Client and DNS Lookup

TOR client is the mandatory component of the Dark Web crawler. It's one of the components that makes our Dark Web crawler from Clear Net Crawler. This component enables the crawler to resolve and fetch TOR based DarkWeb address into their corresponding HTML Web pages. Design a crawler for DarkWeb on TOR network requires additional times for the URL and pages to get resolved into their DNS names. And we believe this factor will hinder the performance of the designed Dark Web crawler. Therefore, we used a DNS cache for fetches URL or dark web domains that are already visited. The DNS lookup component reduced the bandwidth requirement of the network and increases performance of crawler threads and throughput.

4.4 Summary

The chapter systematically went through the designing of Dark Web crawler. The designed system encompasses a number of components working in collaboration to perform the different tasks like crawling, downloading and language identification. In the first phase, a seed URL is feed to the system. In the second phase, the seed URL will be used as URL queue in main memory. In the third phase, extraction of URLs from web link is proceed and put in to the URL queue as shared memory variable, it is accessed by the parallelism of threads using a fork-join thread manager. In the fourth phase, individual crawler thread instance will be forked from the extracted URL Queue, later on put into database when link get visited. In fourth phase, the download manager fetch links from database and fetches the HTML content of individual links using a download thread manager. Instance of single thread is assigned to download a single URL from the total number of 10 download threads used. In six phase, fetched document will be downloaded are saved as html files. Removing of some unwanted details to save disk space is done by an HMTL compressor component. Finally, identification of web documents using the modified G2LI identifier is carried out into a local repository of downloaded web documents. The identification result will be kept into a language identifier database as final output of the proposed system. This enables to query on the database to find out how much of the downloaded Dark Web content is written in Amharic or English.

Chapter Five: Experiment

5.1 Introduction

In this Chapter, we described the implementation details of the architecture of Dark Web crawler. The development environment and the tools used to develop the system are briefly discussed. The tools used for network setup, link and content extraction, downloading and compressing HTML web-pages and with the designed fork-join recursive crawling algorithm are also described. The techniques used and the configurations made on the tools are described in detail. The implementation details are also depicted as screen shots.

5.2 Data Sets

In order to test the designed crawler for DarkWeb search engine we used different .onion websites (Dark Web) based on the purpose and objective of their establishment. All the website we have used were legal and represents a common public interest. Out of the selected 30 datasets, 7 of them are Amharic contents.

All the selected datasets are based on the following characteristic:

- ✓ Legal Website content
- ✓ Amharic content that focuses on security related issues aimed at helping user to be aware on security threats.
- ✓ Social networking Sites
- ✓ News and Radio Websites
- ✓ Can be used in similar way like Clearnet

The selected datasets objectives and their establishment (purpose) is listed in Table 5.1.

Table 5.1: *Data Set Used*

URL_ID	Darknet URL (.onion) Tested	Purpose
1.	http://libraryqtlpitkix.onion/library/	EBook Library
2.	http://sonntag6ej43fv2d.onion/	Benjis's Blog
3.	http://7hk64iz2vn2ewi7h.onion/	Blog about Stories
4.	http://tigas3l7uusztiqu.onion/	Mike Tigas's Blog
5.	http://mpf3i4k43xc2usxj.onion/	Sam Whited Blog
6.	http://fr33tuxnvcaptbam.onion/#blog	Blog
7.	http://bgaxaar7xx6dpptt.onion/	Martine's Blog about History
8.	http://zupyv3e5spdok6nw.onion/cgi-bin/dlbase.cgi?dir=/ebooks	EBook Share and Download
9.	http://libraryqtlpitkix.onion/library/	Jotunbane's Reading Club
10.	http://xfmro77i3lixucja.onion/	The Imperial Library
11.	http://twlba5j7oo5g4kj5.onion/	Free Image Hosting
12.	http://cip5p52lqmufd3kc.onion/	Free File Hosting
13.	http://76qugh5bey5gum7l.onion	Deep Web Radio
14.	http://hxnibog5m2ocjeef.onion/	Deep Web Miniseries (*Shinning the Light of the Gospel in the Deep Web*)
15.	http://scihub22266oqcxt.onion/	First website in the world to provide mass & public access to research papers
16.	http://maqc6rgyz3pnoteh.onion/ am/avast	Amharic (አቫስት! ጸረ ቫይረስ)
17.	http://maqc6rgyz3pnoteh.onion/am/eraser	Amharic (አስተማማኝ የፋይል ማስወገጃ)
18.	http://maqc6rgyz3pnoteh.onion/am/chapter-6	Amharic (ስሱ መረጃዎችን በጥንቃቄ ማጥፋት)
19.	http://maqc6rgyz3pnoteh.onion/am/chapter-10-3	Amharic (ማኅበራዊ መረቦች/ገጾች)

20.	http://maqc6rgyz3pnoteh.onion/am/riseup	Amharic(አስተማማኝ የኢሜይል አገልግሎት)
21.	http://maqc6rgyz3pnoteh.onion/am/pidgin	Amharic(አስተማማኝ የፈጣን መልእክት)
22.	http://maqc6rgyz3pnoteh.onion/am/vaultletsuite	Amharic(ደኅንነቱ የተጠበቀ የኢሜይል አገልግሎት)
23.	https://www.facebookcorewwwi.onion/	Dark Web's Facebook Version
24.	http://blkbook3fxhcsn3u.onion/index.php/	TOR Social Networking Community
25.	http://sinbox4irsyaauzo.onion/	Anonymous Email Services
26.	http://oxwugzccvk3dk6tj.onion/	8 Chan (*you can create your own image board for free with no experience or programming knowledge needed)*
27.	http://7rmath4ro2of2a42.onion/	News Media
28.	http://w5ifkainqlgtvg7a.onion/	Live Radio News
29.	http://dslsjffqyhv5vdx2.onion/ENTER/music/	Music Listening Service
30.	http://n3q7l52nfpm77vnf.onion/	History Archives

5.3 Implementation

The prototype is implemented using Java as programming language, MySQL database to store crawled data and the following tools:

JDK1.8.0_20 with Spring Tool Suite IDE 3.7:

This is the latest version of the Java[1] SDK (Software Development Kit) for developing and writing Java based programs. The software is installed in a Linux environment and different components of the proposed prototypes is developed. It is based on the Java Programming Language. We choose this tool for following reasons:

- A thread concurrency and parallelism support

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

- A strong experience in the language
- The platform independent nature of the language
- The availability of open source components of the language on the web
- The Object oriented feature of the language
- A variety of choice to among available open source support libraries

HTML parser

HTML Parser[2] is a Java library used to parse HTML in either a linear or nested fashion. Primarily used for transformation or extraction, it features filters, visitors, custom tags and easy to use JavaBeans.

Jsoup Parser

Jsoup3 is a Java library for working with real-world HTML. It provides a very convenient API for extracting and manipulating data, using the best of DOM, CSS, and JavaScript library-like methods.

TOR Client

TOR[4] Client software is a free software implementation of second-generation onion routing, a system enabling its users to communicate anonymously on the Internet.

MySQL Database Server 5.5

It is a multi-threaded, multi-user, and Structured Query Language (SQL) database server. We have chosen this software because it is open source, works with our choice of programming language, and support concurrency and connection pooling for database connections made from third party applications.

---

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

[3] http://jsoup.org

[4] *https://www.torproject.org/*

Html Compressor

Html Compressor[5] is a small Java library that minifies given HTML or XML source by removing extra white spaces, comments and other unneeded characters without breaking the content structure. As a result pages become smaller in size and load faster.

Apache DBCP Component

Apache DBCP[6] Component Opening a connection per crawler activity is infeasible in different situations where the number of simultaneous thread count can be very large. This Commons package provides an opportunity to coordinate the efforts required to create and maintain an efficient, database connection.

G2LI

Global Information Infrastructure Laboratory Language Identifier (G2LI)[7]. It is a text language identifier that can be used to simplify on how to understand the type of language, script and encoding schemes a text content is written in.

NetBeans IDE

NetBeans[8] is a community supported open source integrated development environment for developing Java platform application. Net beans 8.1 is used to implement the user interface of the language identifier, for decompiling G2LI byte code into standard Java human readable object class, to fix errors and library reference issues on G2LI and design the Dark Web crawler UI.

We used single laptop computers for the implementation, testing and evaluation purpose. The computer has Intel(R) Pentium(R) Dual Core CPU with 2.00 GHz processor, 4.00GB

[5] https://code.google.com/p/htmlcompressor/

[6] http://apache.org/

[7] http://www.ijcsi.org/articles/Language-Identification-of-Web-Pages-Based-on-Improved-Ngram-Algorithm.php

[8] https://netbeans.org

of RAM, 320GB of hard disk, and 64 Bit Linux operating system. We used 0.48 Mb/s network bandwidth.

The prototype is initialized by using a seed URL as it is shown on the graphical user interface shown in Figure 5.1. The designed system crawled 67,602 Dark Web URLs and downloaded 56,304 dark web documents and 13,000 Hidden Services within a two weeks of crawling for at least 12 hours a day starting from October, 2015. During the crawling process, downloading process is also running on the same single machine. The downloaded web documents sized 800 MB.

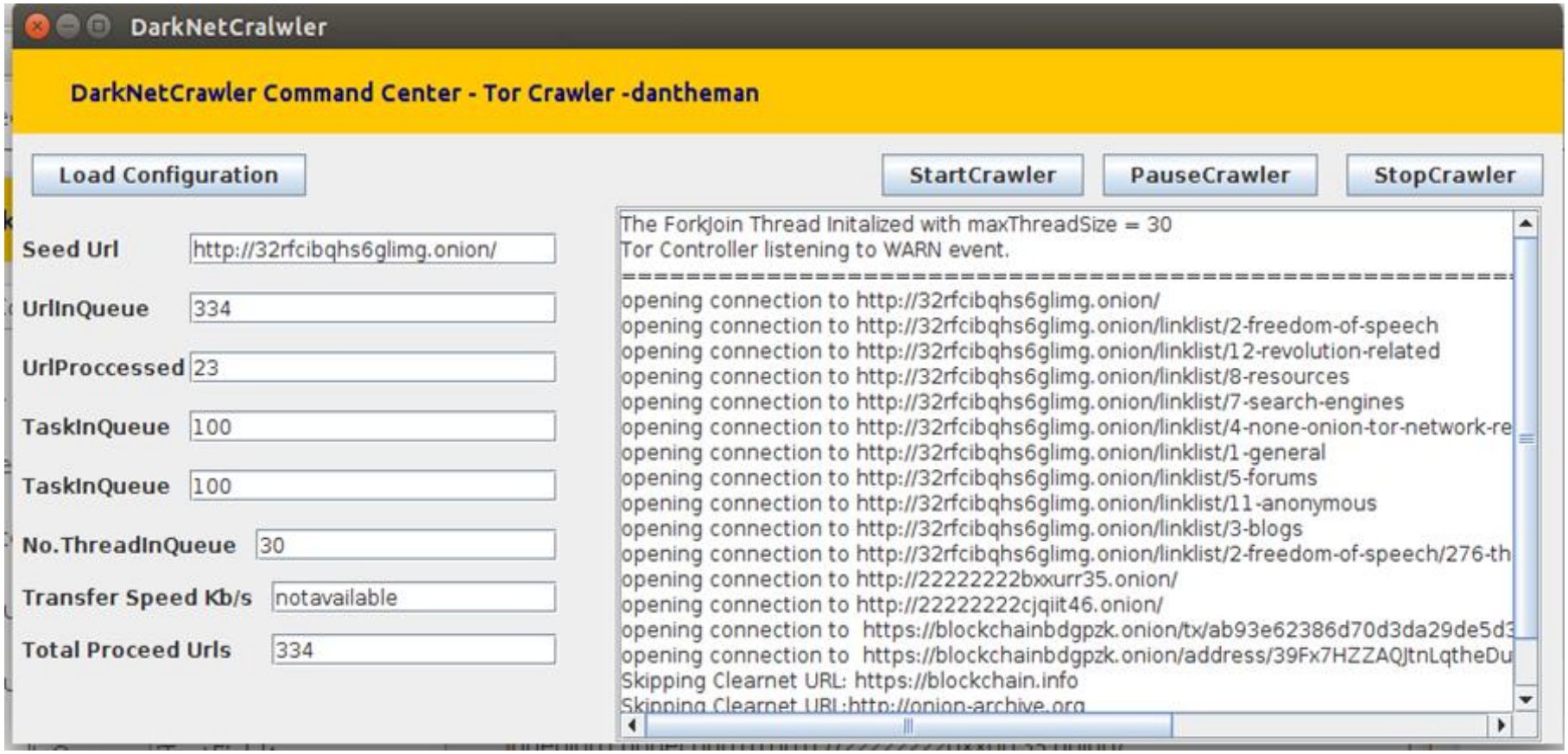


Figure 5.1: *Running Prototype for Dark Web Crawler*

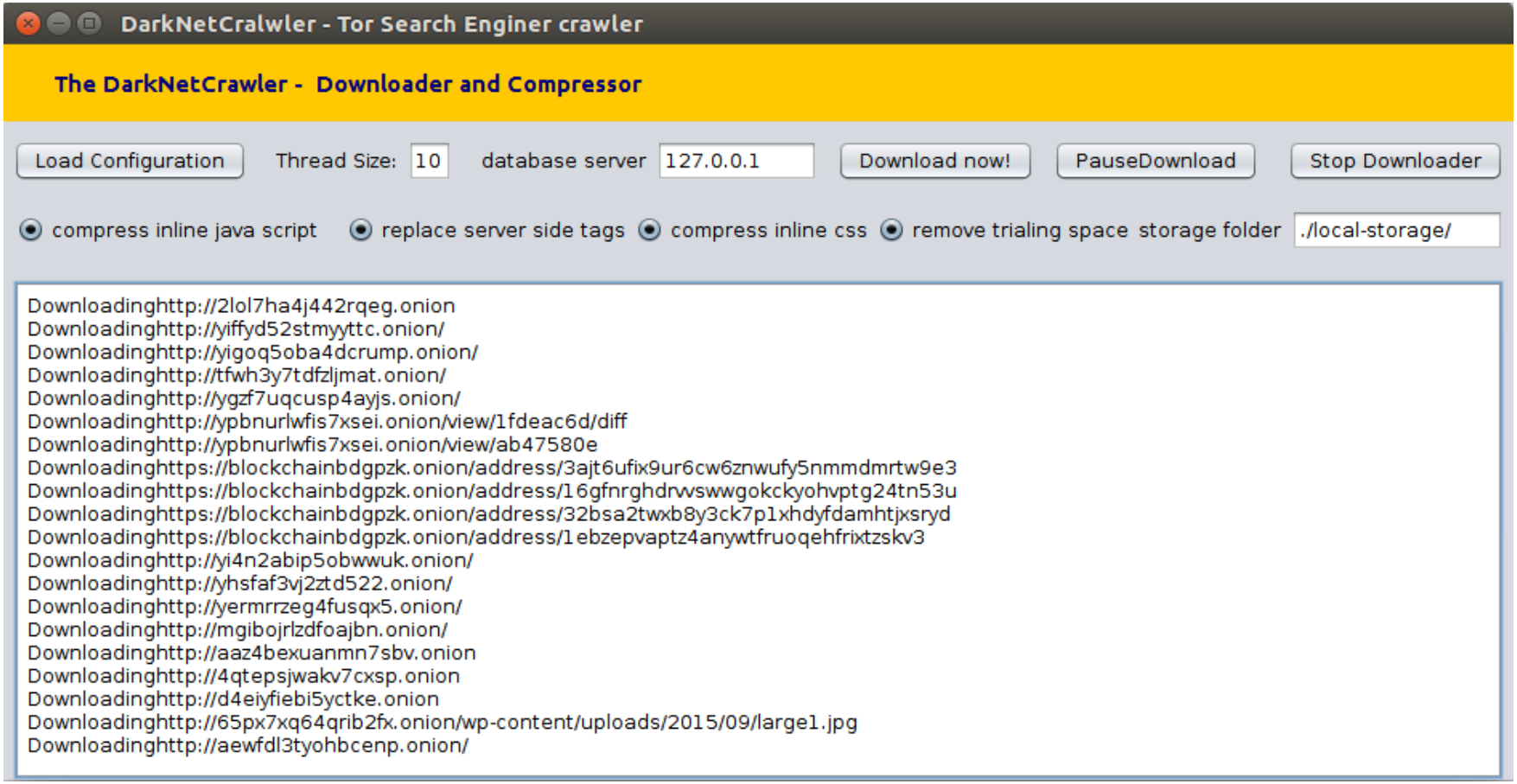


Figure 5.2: *Download Manager and HTML Compressor User Interface*

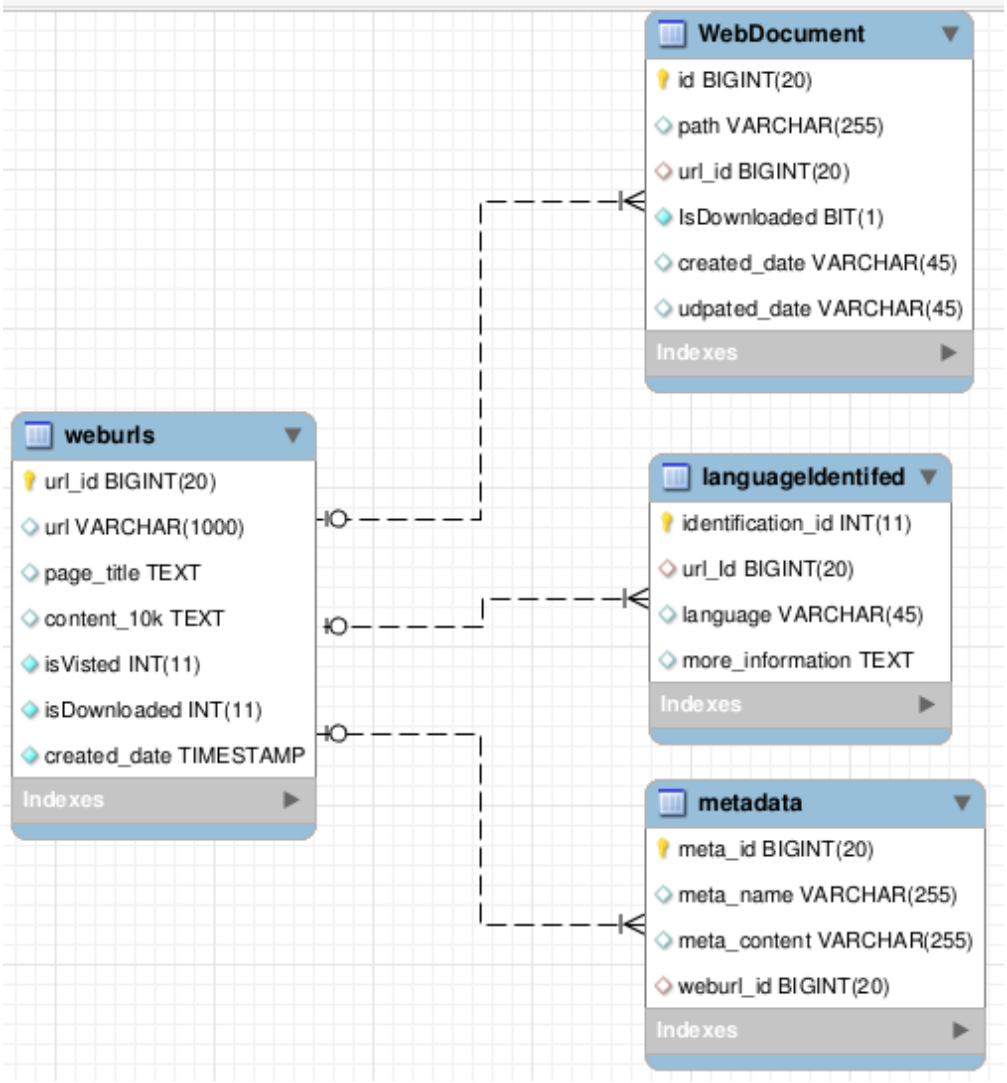


Figure 5.3: *Conceptual Database Schema Definition of a Dark Web Crawler and Offline Language Identifier*

Download Manager

A Download Manager process is separate from the proposed Dark Web crawler process. This means the two are integrated using a shared database link but they are not dependent to each other to do their individual tasks. The download manager can even be configured to work in different machines than the dark web crawler is deployed. The Download manager use the connection pooling approaches provided by Java's Executor Services framework. We have used a Max Thread Pool Size of 10. Figure 5.2 and Figure 5.4 show, the developed user interface for a download manager with a sample dark website downloaded, respectively. Other downloaded dark websites are given on Appendix 1-5.

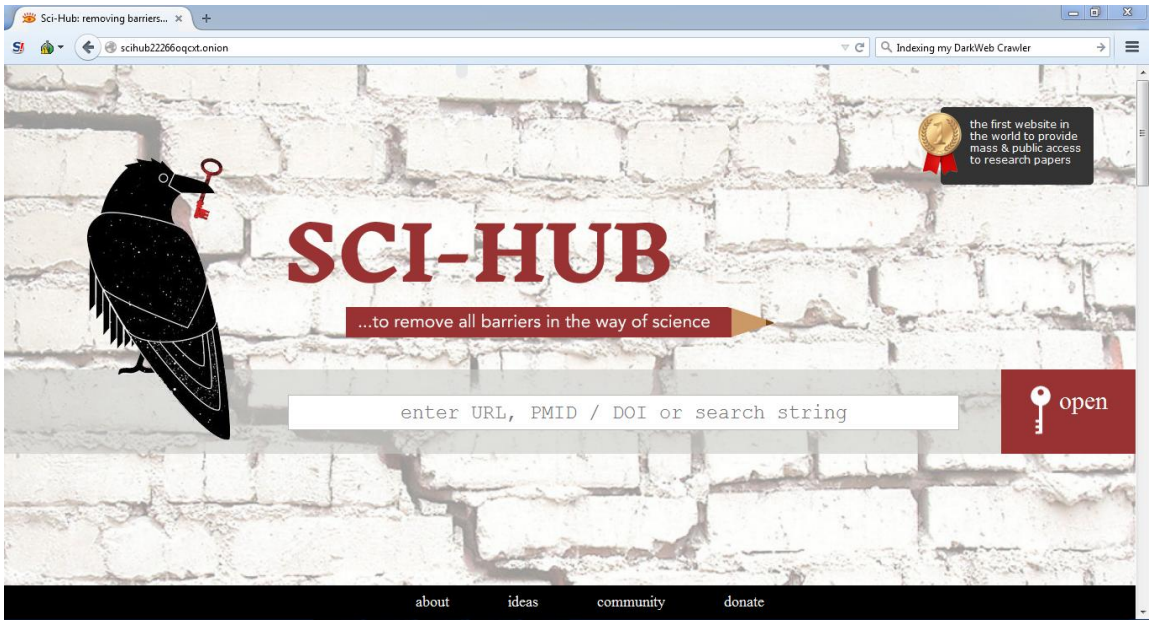


Figure 5.4 *: Sample Dark Net Website Downloaded*

Html Compressor

The HMTL compressor implementation has been adopted from a Google's compression code using the HTML compressor Library for Java. The Html compression set different configuration for optimum compression size of a given Dark Web site when it is downloaded. After compressing and saving the HTML document into a file, individual URLs link status is updated on the database (*i.e.*, it change the download Status to true). Later when other thread instances access the URL it will not be available for download.

Implementation of Offline Language Identifier

We have adopted the G2LI. G2li is an n-gram based language identifier and it is selected due its high accuracy in identification, its resilience to typographical errors, and its minimal data requirement for training data. This means by using n-gram sequence of bytes it can identify a language using a combination of bytes from a training data. In this study we have adopted the G2LI by adding some modification to work as per our requirement .The implementation detail and customization that are made on G2LI is discussed in Chapter Four, Section 4.3.6. The screenshot for the prototype of an offline language identifier is shown in Figure 5.5.

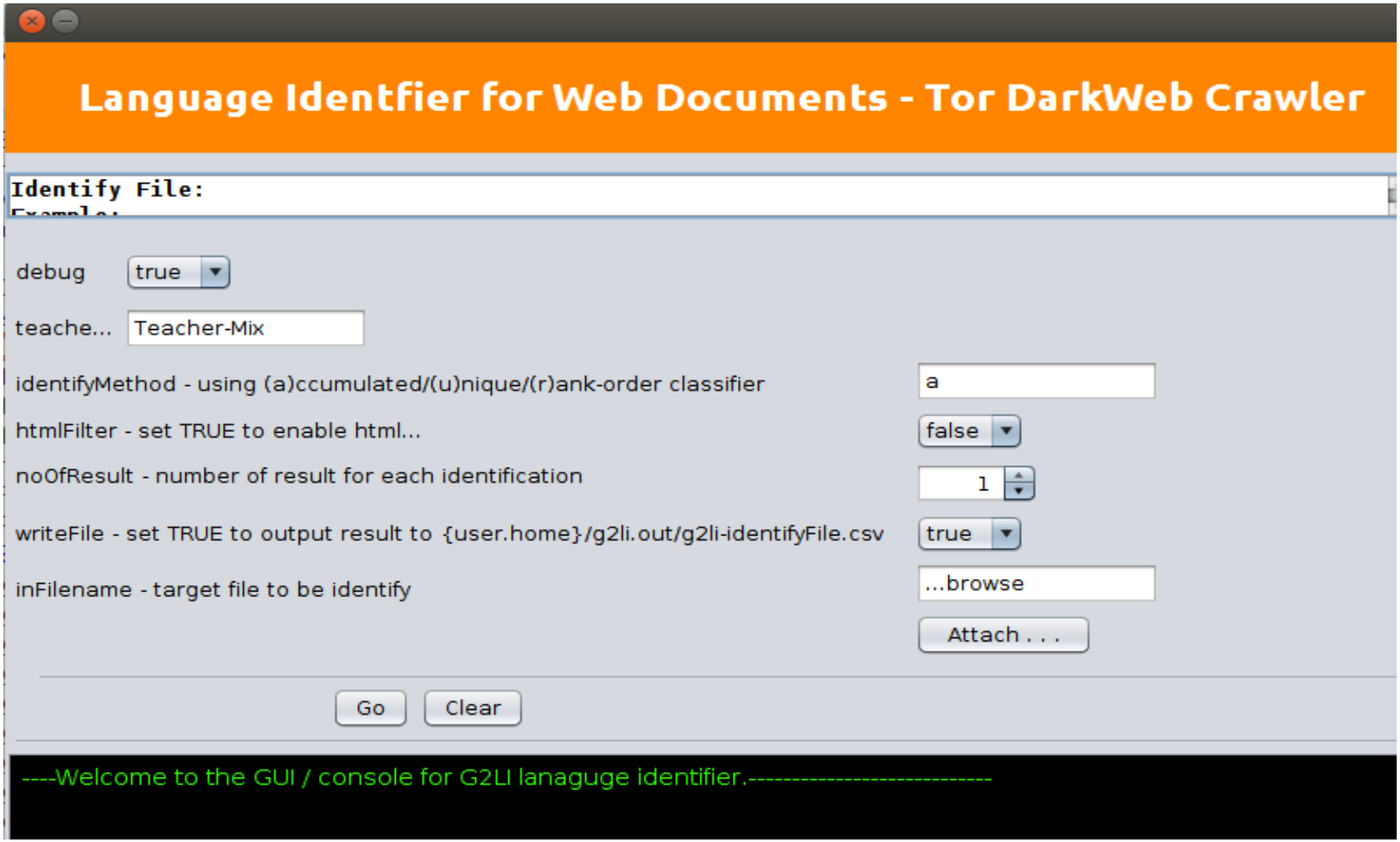


Figure 5.5: *Running Prototype for Language Identifier*

5.4 Test Result

The objective of this research is to design a Dark Web crawler which will be used to crawler the TOR network and make contents visible to end users. What we have tested is from the downloaded dark web documents and crawler database we compare our findings with a Clearnet search engine.

In testing the designed Dark Web crawler, we considered the following parameters to compare our finding with the Google search engine crawler.

- A page title extracted from a Dark Web site
- A meta tag keyword extracted from Dark Web site
- A Dark Web site's content from its home page and other child pages

Table 5.3 shows the selected datasets URLs (.onion website). Also, the page tile and Meta tag information found on the selected dataset is given. The No. column indicates the reference number given to selected onion URLs. The Second column show the page title extracted from the selected URL and third column show the Meta Tag information of these selected onion sites. The test result from a Dark Web crawler is shown on Table 5.4. In Table 5.4, the selected dataset are searched from the Dark Web crawler database and file storage based on the parameters selected on Table 5.2 and Table 5.3. Later, a search on each dataset is performed using the parameters as shown on the Table 5.3 (page tile and meta-tag) above using Google search engine (Clearnet Search Engine) and Table 5.4 shows, the results found.

Table 5.2*: Test result of the designed Dark Web Crawler*

URL_ID	**Page Title Found**	**Meta Data found**
1	*Index of /library/Computing/*	*No Meta Info*
2	*Benjamin Sonntag –Benj's blog !*	*No Meta Info*
3	*Best Blog Stories*	*No Meta Info*
4	*Mike Tigas*	*Mike Tigas is a developer, journalist, photographer, civic hacker, and security/privacy tinkerer at ProPublica.*
5	*SamWhited.com|blog*	*The personal blog of Sam Whited*
6	*Jérôme.B; Technology, Travels, Free Culture...*	*No meta Info*
7	*Lapsed Ordinary | Martijn's blog*	*No Meta Info*
8	*Datenhangar - Directory Listing Script*	*damas,virii,dlbase,dlbaze,downloadbase,Datenhangar*
9	*Index of /library/*	*No Meta Info*
10	*Imperial Library of Trantor*	*No Meta Info*
11	*Image Hosting*	*No Meta Info*
12	*Free File Hosting*	*No Meta Info*
13	**** Deep Web Radio ****	*No Meta Info*
14	*Deep Web Ministries Index page*	*No Meta Info*
15	*Sci-Hub: removing barriers in the way of science*	*The first pirate website in the world to open to mass and public access to tens of millions research papers*
16	*Security In A Box | Tools and tactics for your digital security*	*Security in-a-box is a collection of guides and free tools to secure your computer …*
17	*Security In A Box | Tools and tactics for your digital security*	*Security in-a-box is a collection of guides and free tools to secure your computer …*
18	*Security In A Box | Tools and tactics for your digital security*	*Security in-a-box is a collection of guides and free tools to secure your computer …*
19	*Security In A Box | Tools and tactics for your digital security*	*Security in-a-box is a collection of guides and free tools to secure your computer …*

20	*Security In A Box | Tools and tactics for your digital security*	*Security in-a-box is a collection of guides and free tools to secure your computer …*
21	*Security In A Box | Tools and tactics for your digital security*	*Security in-a-box is a collection of guides and free tools to secure your computer …*
22	*Security In A Box | Tools and tactics for your digital security*	*Security in-a-box is a collection of guides and free tools to secure your computer …*
23	*Facebook*	*No Meta Info*
24	*TOR Social Networking Community*	*No Meta Info*
25	*Sinbox*	*No Meta Info*
26	*8chan, the infinitely expanding imageboard*	*No Meta Info*
27	*SoylentNews: SoylentNews is people*	*No Meta Info*
28	*World News Radio Today*	*World News | Technology | Science | Politics | Talk Back Radio | Gaming*
29	*/ENTER/music*	*No Meta Info*
30	*History Archive*	*The most complete library of historical data on Marxism and the revolutionary working class.*

Table 5.3: *Test Result from Google Search Engine Crawler*

DarkNet URL_ID	***Page Title found***	***Meta tag found***	***Status***
1	*www.ipl.org/IPLBrowse/GetSubject?vid=13...* *https://en.wikipedia.org/wiki/Library_(computing)* *https://en.wikipedia.org/wiki/Library_computer_system*	*Url has No Meta*	*Not Crawled*
2	https://benjamin.sonntag.fr/en https://benjamin.sonntag.fr/ https://benjamin.sonntag.fr/ml/azov	*Url has No Meta Data*	*Not Crawled*
3	*https://www.quora.com/What-are-the-best-short-story-blogs* *https://longreads.com/* *www.veryshortfiction.com/*	*Url has no meta*	*Not Crawled*
4	*https://twitter.com/mtigas* *https://www.propublica.org/site/author/mike_tigas* *https://github.com/mtigas*	*https://mike.tig.as/* *https://mike.tig.as/about/* *alcohol-rehab.seoweasel.com/www.mike.tig.as*	*DarkWeb Version Not Crawled*
5	https://blog.samwhited.com/ https://blog.samwhited.com/about https://blog.samwhited.com/2015/05/writing-of-ships	*https://blog.samwhited.com/* *https://blog.samwhited.com/about* *https://blog.samwhited.com/2013/07/dont-be-evil*	*DarkWeb Version Not Crawled*

6	*https://fr33tux.org/* *https://fr33tux.org/post/a-new-pad/* *https://fr33tux.org/pages/me/*	*Url has No meta Info*	*Not Crawled*
7	*https://www.lapsedordinary.net/* *https://www.lapsedordinary.net/...* *https://www.lapsedordinary.net/category/security/*	*www.homeenglish.ru/Books.htm* *frenglish.ru/start_reading.html* *https://itunes.apple.com/lv/app/...*	*Not Crawled*
8	*https://download.adamas.ai/cgi-bin/dlbase.cgi?dir=/Scripts* *https://download.adamas.ai/* *https://download.adamas.ai/cgi-bin/dlbase.cgi?dir=/ezines*	*https://download.adamas.ai/* *https://www.reddit.com/rthe_hidden_internet_riseup_dnf_datenhanga* *https://sourceforge.net/projects/datenhangar*	*Not Crawled*
9	*www.ncl.ac.uk/library/resources/new_books/* *holtz.org/Library/* *https://sin.thecthulhu.com/library/*	*Url has no meta info*	*No result from Darknet version of Facebook*
10	*https://en.wikipedia.org/wiki/Library_of_Trantor* *https://www.reddit.com/r/.../imperial_library_of_trantor_an_onion_epub.* *https://www.reddit.com/r/onions/comments/*	*Url has no Meta Info*	*Not Crawled*
11	*postimage.org/* *imgur.com/* *https://imgsafe.org/*	*Url has No meta*	*Not Crawled - Darknet Version is hidden*

12	*www.filedropper.com/* *www.tinyupload.com/* *thetechreader.com/.../15-top-free-file-hosting*	*Url has No meta*	*No result from Darknet Version*
13	*https://www.youtube.com/watch?v=7lFGkvgl2u4* *https://www.reddit.com/.../im_new_to_tor_and_deep_web* *https://www.mixcloud.com/discover/deep-web/*	*Url has No meta*	*Not Crawled*
14	*the-hidden-wiki.com/* *hxnibog5m2ocjeef.onion.glass/* *www.deepweb-sites.com/deep-web-links-2015/*	*Url has no meta*	*Not Crawled*
15	*scihub22266oqcxt.onion.link/* *https://sci-hub.io/* *https://news.ycombinator.com/item?id=11093779*	*www.sciencealert.com/* *www.independent.co.uk* *https://bluesyemre.com/.../sci-hub-the-first-pirate*	*Not crawled*
16	*https://securityinabox.org/en* *https://securityinabox.org/en/tactics* *https://tacticaltech.org/*	*https://securityinabox.org/en* *https://www.privacyrights.org/securing-your-computer-maintain-your-pr.* *https://oag.ca.gov/privacy/facts/*	*Not Crawled*
17	*https://securityinabox.org/en* *https://securityinabox.org/en/tactics* *https://tacticaltech.org/*	*https://securityinabox.org/en* *https://www.privacyrights.org/securing-your-computer-maintain-your-pr.* *https://oag.ca.gov/privacy/facts/*	*Not crawled*

18	*https://securityinabox.org/en* *https://securityinabox.org/en/tactics* *https://tacticaltech.org/*	*https://securityinabox.org/en* *https://www.privacyrights.org/securing-your-computer-maintain-your-pr.* *https://oag.ca.gov/privacy/facts/*	*No crawled*
19	*https://securityinabox.org/en* *https://securityinabox.org/en/tactics* *https://tacticaltech.org/*	*https://securityinabox.org/en* *https://www.privacyrights.org/securing-your-computer-maintain-your-pr.* *https://oag.ca.gov/privacy/facts/*	*Not Crawled*
20	*https://securityinabox.org/en* *https://securityinabox.org/en/tactics* *https://tacticaltech.org/*	*https://securityinabox.org/en* *https://www.privacyrights.org/securing-your-computer-maintain-your-pr.* *https://oag.ca.gov/privacy/facts/*	*Not Crawled*
21	*https://securityinabox.org/en* *https://securityinabox.org/en/tactics* *https://tacticaltech.org/*	*https://securityinabox.org/en* *https://www.privacyrights.org/securing-your-computer-maintain-your-pr.* *https://oag.ca.gov/privacy/facts/*	*No crawled*
22	*https://securityinabox.org/en* *https://securityinabox.org/en/tactics* *https://tacticaltech.org/*	*https://securityinabox.org/en* *https://www.privacyrights.org/securing-your-computer-maintain-your-pr.* *https://oag.ca.gov/privacy/facts/*	*Nor Crawled*

23	*https://www.facebook.com/* *https://twitter.com/facebook?lang=en* *https://play.google.com/store/apps/details*	*Url has no meta data*	*Not Crawled*
24	*culturedigitally.org/2014/10/social-networking-on-the-dark-web/* *https://blog.torproject.org/category/tags/social-media* *https://www.reddit.com/.../TOR/.../networkonion*	*https://www.scribd.com/doc/175604864/Deep-Web-Links* *www.dailydot.com/.../deep-web-silk-road-black-market-re...* *boingboing.net/2011/01/05/in-vogue-magazine-6-.html*	*Not crawled*
25	*https://www.youtube.com/watch?v=rNT-L9nSCG0* *https://www.youtube.com/watch?v=rRKNnhP2sS8* *https://www.youtube.com/watch?v=0Md5US0ktyI*	*Url has no Meta Data*	*Not crawled*
26	*oxwugzccvk3dk6tj.onion.link/* *linkis.com/8chan.co* *https://8ch.net/boards.html*	*Url has no Meta Data*	*Not crawled*
27	*https://soylentnews.org* *https://soylentnews.org/meta/* *https://soylentnews.org/index.pl?issue=20160410*	*Url has no Meta Data*	*Not Crawled*
28	*www.worldnewsradio.today/* *tunein.com/radio/World-News-g3125/* *tunein.com/radio/World-News-Radio-Today-s231827/*	*www.npr.org/* *www.newstalkzb.co.nz/* *talksport.com/*	*Not Crawled*

29	*https://pro.beatport.com/label/enter-music/39215* *www.facebook.com* *enterexperience.com/*	*Url has no meta Data*	*Not Crawled*
30	*https://archive.org/web/* *https://en.wikipedia.org/wiki/Archive* *https://sfi.usc.edu/vha*	*www.trotsky.net/revolution_betrayed.html* *https://www.facebook.com/ukuncut/photos/a../947713935276535/* *https://en.wikipedia.org/wiki/Influences_on_Karl_Marx*	*Not Crawled*

According to the result of the experiment, there are 30 datasets selected onion websites, as shown in Table 5.1. Among the selected 30 dataset, the proposed Dark Web crawler found and downloaded 30 of the onion website URLs. As shown in Table 5.2 the proposed system has found the page titles, Meta tags and downloaded contents of the 30 selected datasets for later search and retrieval process. Whereas, the Google search engine failed to find the onion websites using page title, Meta tag and sample content taken from the data sets from the top 10 search results retuned.

5.5 Discussion

Information Retrieval systems like search engines use crawlers to discover contents from the Web. They systematically index web documents and keep them as inverted index. The indexing of terms help to retrieve documents related to search term when user types search word using a query engine. Search result should reflect what has been indexed and results should be retrieved accordingly. This is measured by precision and recall over the retrieved documents. When such conditions not meet the information retrieval process fails. Table 5.4, summarizes the total findings of the proposed system.

Table 5.4 : *Summary of results from Dark Web Crawler*

Category	**Total Found**
DarkWeb site crawled	67602
Hidden Service or DarkWeb Hosts	13000
DarkWeb sites having a page title	66031
DarkWeb sites with meta data	22196
DarkWeb site downloaded	56304
Amharic Language Contents	7

Clearnet search engines face the problem of exposing content to the Clearnet, because they are not design to do so architecturally. Since they should make result available to be used by all user of the Web. Net content by just clicking on resulted links (*i.e.*, assuming if they have crawled and indexed). If these search engine provided the result user also require additional software like TOR browser to access the contents, which is different from the usual case on Clearnet search and retrieval process. According to this thesis findings, most hidden service website does not describe their website using a Meta tag, which is often the case Clearnet search engine like Google use to index websites and this leads to being not covering the Dark Web content. We suggest that they should use additional parameters like contents to index and crawl the Dark Web. According to this study, we showed that the problem of Clearnet search engine not crawling and indexing content inside the DarkNet, typically on TOR network. This is true when a search engines are not crawling

content by accessing TOR network to get Dark Web contents and remains uncovered. This requires a modification on their architecture.

As this work is the first in designing a Dark Web crawler for crawling dark web sites and identification of Amharic dark web contents, we cannot compare the performance with other attempts. Previous attempts focused only on a few and selected dark website crawling, and does not even include language identification in their crawler architecture. The previous researches reviewed on Chapter Three commonly were not capable of crawling the entire TOR network (DarkNet), lack to have description of architectural model of crawlers when used, used 1-4 dataset to do content analysis, aimed at only text and topic classification than surfacing web content to end users, many of them focused on traffic analysis, and lack to design an integrated linguistic analyzer for identification of dark web documents available in Amharic languages. But, in our study we have designed an architecture for a Dark Web crawler to that it can be used to search if indexer and query engine components is added to it.

The heart component of a Dark Web is its crawler and the rest of the components like indexer and query engine can be applied with similar nature and technique like that of a Clearnet search engine (see Chapter Two –Section 2.3.6). The integration of these components does not require further architectural adjustment to work with the proposed Dark Web Crawler. Indexer and Query engine components with same design for Clearnet can be used. This will be a future objective to realize a complete search engine system for DarkWeb.

In summarization of our experimentations, Table 5.4 shows the overall finding using the proposed design and according to the finding we have crawled over 13,000 hidden services, 67602 individual dark web URLs discovered, 56304 DarkNet web documents downloaded resulting an 800 MB data. More storage disk space was required if we had not deployed an HTML compressor component to reduce the size of different downloaded dark web sites. Finally, we took a sample from our crawled data set and we have showed the result with that of Google search engine. According to the experiment the selected dark web sites were not able to be found on using a Clearnet search engine (Google).

Chapter Six: Conclusion and Future Work

6.1 Conclusion

The Internet has been the source of information for heterogeneous data types. TOR is used to establish anonymous communication using the Onion Route protocol on the Internet. TOR has a comprehensive collection of different information isolated from the Clearnet. Its source of information are organized in different web documents deployed under anonymous hidden services. This web contents are written in different languages. According to [57], TOR usage is increasing over time .The amount of information on the network which is being kept is also growing. This in turn; result as huge repository of web data to be harvested in the future from dark web, alone. In addition, researches should not only being focused on TOR improvement of its privacy and security model, but should give emphasis to surfacing dark web contents to end users. In doing so, TOR's anonymity and privacy model should be respected (*i.e.,* publishing and accessing content anonymously).

Although a few works have attempted to study the actual content characteristics residing on the TOR network. Furthermore, studies on how to make available contents from the dark web and provide a model of a search engine remains uncovered. On this work done so far lack to follow a systematic approach to solve the problem of making contents available to wider public on Clearnet. Even more, attention has not been given in prospect to design a Dark Web crawler that can serve as automatic information retrieval means. In relation, no work has attempted to address this issue of Amharic content identification residing in this network.

In order to address the objectives of this study it is necessary to examine the techniques and technologies for design of Dark Web crawler. A comprehensive review of the state of the arts in the Web IR was considered. After understanding of these techniques and technologies allowed the design for requirements for each component of a Dark Web crawler. The proposed Dark Web crawler consist of crawlers that work under a parallel and recursive algorithm called Fork-Join , a language identifier for identification of web documents and download manager components. These components were suggested for the first time on this study, targeting to be used in TOR network as Dark Web crawler. The evaluation took from the development of a demonstration application and proof of concept

tests (*i.e.*, comparison of result with the Google search engine) from the top 10 returned results. The success of the demonstration and proof of concept tests clearly shows the feasibility of providing a search engine system for crawling a TOR network to surface its content.

6.2 Contribution of this Work

The overall study has the following contributions:

- Identified appropriate crawling algorithm, approach and adopt to fit to the requirement.
- Developed an algorithm that can crawler the entire Dark Web using on a Fork-Join method.
- Provides a model of a Dark Web crawler and Downloader with integrated language identifier for Amharic.
- Developed an algorithm that can identify Dark Net URLs while crawling.
- Developed an algorithm that can extract Links, Page Title and Meta information from Dark Net website while crawling.
- Developed a system for Web content categorization by topic and by language.
- Handled weak spot of the TOR network when crawling to address issues in further crawling attempts.
- Provided a Dark web address filter for handling dark web URLs when crawling.
- Suggested improvements for communication by using DNS cache client to be embedded with a TOR client software.
- Demonstrated a compression mechanize to reduce disk space when storing downloaded web documents, helped to store large size data into small size storage.
- Identify appropriate language identifier tool modified it to allow UI based command system over a command line, and to store identification information on database.
- Developed a Dark Web crawler system that be integrated to an existing Indexer and Query engine components of a search engine.
- We provided a Tool for future researchers aimed at Dark Web who are interested in content based research to collect and gather dark web documents from TOR Network.

6.3 Future Work

In this research, we designed and developed a Dark Web crawler that work under the TOR network's protocol.

Although the system had already demonstrated a good performance on a current environment, there is still room for improvement. As a recommendation of improvement to our designed model, the following need to be incorporated in future work:

- Adding a DNS cache client with a software based implementation which has been provided by the MIT license to improve the speed of the crawler.
- Additional detection of TOR client software when it gets trapped by too many web request and reload it by freezing all the active crawling thread and resume later.
- To make the crawler threads distributed and scalable
- Integrate Index and Query engine to make it a full-fledged search engine to query over data.

References

[1] http://www.internetworldstats.com, "The Internet World Stats," [Online]. Available: http://www.internetworldstats.com/emarketing.htm. [Accessed 20 May 2015].

[2] http://www.internetworldstats.com, "The Internet World Stats," [Online]. Available: http://www.internetworldstats.com/stats.htm. [Accessed 29 Feb 2016].

[3] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag "Anonymous Connections and Onion Routing," *IEEE Journal on Selected Areas In Communications,* Vol. 16, 1998.

[4] Roger Dingledine, Nick Mathewson, and Paul Syverson, "Tor: The Second-Generation Onion Router," *In the Proceedings of USENIX Security Symposium,* Vol. 13, 2004.

[5] Zzz (Pseudonym) and Lars Schimmer, "Peer Profiling and Selection In the I2P Anonymous Network," *In the Proceedings of PET-CON* , Dresden, Germany, March 2009.

[6] Biryukov Alex, Ivan Pustogarov, and R. Weinmann, "Trawling For Tor Hidden Services: Detection, Measurement, Deanonymization," *IEEE Symposium in Security and Privacy (SP),* 2013..

[7] Haraty R.Atsu, Zantout B., "The TOR Data Communication System," *Journal of Communications and Networks,* Vol. 16, No. 4, pp. 415 - 420, 2014.

[8] Sonali Gupta and Komal Kumar Bhatia, "Comparative Study of Hidden Web Crawlers," *International Journal of Computer Trends and Technology* , Vol. 12, June 2014.

[9] M. K. Bergman, "White Paper: “The Deep Web: Surfacing Hidden Value,” *Journal of Electronic Publishing,* Vol. 7, No. 1, 2001 .

[10] Alex Biryukov , Ivan Pustogarov and Ralf-Philipp, "Content and Popularity Analysis of Tor Hidden Services," in *IEEE 34th International Conference On Distributed Computing Systems Workshop,* Madrid, June 30, 2014-July 3, 2014.

[11] Christopher D.Manning , Prabhakar Raghavan , and Hinrich Schutze, An Introduction to Information Retrieval, England : Cambridge University Press , Online Edition , 2009 .

[12] E. Greengrass,"Informational Retrieval :A Survey",University of Maryland," Baltimore County , November 2000.

[13] O. McBryan, "GENVAL and WWWW: Tools for Taming the Web," *International World Wide Web Conference,* 2004.

[14] Vladislav Shkapenyuk and Torsten Suel,"Design and Implementation of a High Performance Distributed Web Crawler," *In Procceding of the Int. Conf.on. Data Engineering*, 2002 .

[15] S. Chakrabati, Mining the Web ,Discovering Knowledge form Hypertext Data, Morgan Kaufmann Elsevier Science , CA .San Francisco , USA , 2003.

[16] Apoorv Vikram Singh , Vikas and Achyut Mishra, "A Review of Web Crawler Algorithms," *(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (5) , 2014*.

[17] Pierre Baldi , Paolo Frasconi and Padhraic Smyth, "Modeling the Internet and the Web Probabilistic Methods and Algorithms", Wiely & Sons Ltd, 2003.

[18] Y. L. Iarapakis , "SlideShare.com," [Online].Available: http://www.slideshare.net/iarapakis/upf15. [Accessed October 2015].

[19] Rashmi Janabdhu ,Prashant Dahiwale and M.M.Raghuwanshi, "Analysis of Web Crawling Algorithms," *International Journal on Recent and Innovation Trends in Computing and Communication* , vol. 2, no. 3, pp. 488-492.

[20] Allan Heydon and Marc Najork, "Mercator : A Scalable ,Extensible Web Crawler," *Journal of World Wide Web Conf,* 1999.

[21] S. Brin and L. Page, "The Anatomy of a Large-Scale Hyper-Textual Web Search Engine," *In Procceding of the 7th World Wide Web*, 1998.

[22] Christopher Olson and Marc Najork, "Web Crawling," *Foundations and Trends in Information Retrieval ,* Vol. 4, pp. 175-246, 2010 .

[23] A. Shen, "Algorithm and Programming Problems and Solution", Springer , 2nd ed. 2010 , pp.135.

[24] Monica Peshave and Kamyar Dezhogosha, "How Search Engines Work and a Web Crawler Application," University of Illinois,, Springfield USA, 2015.

[25] M. K. Bergman, "The Deep Web: Surfacing Hidden Value," *Journal of Electronic Publishing*, Vol. 4, No. 3, pp. 175-246, 2010.

[26] A. S. Tanenbaum and R. van Renesse, "Distributed Operating Systems," *ACM Computing Surveys*, Dec. 1985.

[27] TryEngineering, "www.tryengineering.org," Feb 2016. [Online]. Available: http://www.tryengineering.org/lessons/searchengines.pdf. [Accessed Feb 2016].

[28] Doug Lea, "A Java Fork/Join Framework," *ACM Conf. on Java*, 2000.

[29] Umut A. Acar, Arthur Charguéraud, Mike Rainey, "Scheduling Parallel Programs by Work Stealing with Private Deque," *ACM SIGPLAN Notices*, 2013.

[30] Blumofe, Robert D., and Charles E. Leiserson., "Scheduling Multithreaded Computations by Work Stealing," *Journal of the ACM (JACM)*, 1999.

[31] Madhavan J. , David Ko and Lucja Kot, "Google's Deep Web Crawler," *Proceedings of the VLDB Endowment 1.2*, 2008.

[32] Noor, Umara, Zahid Rashid, and Azhar Rauf., "A Survey of Automatic Deep Web Classification Techniques," *International Journal of Computer Application*, Vol. 19, No. 6, April ,2011.

[33] A. Ntoulas, "Crawling and Searching the Hidden Web," *PhD diss. ,University Of California Los Angeles*, 2006.

[34] Mashael AlSabah and Ian Goldberg, "Performance and Security Improvements for Tor: A Survey," *IACR Cryptology ,* ePrint Archive 2015.

[35] Conrad Bernd and Fatemeh Shirazi., "A Survey on Tor and I2P," In *Procceding Of 9th ICIMP ,* 2014.

[36] Tor Project, "Tor Project," Ocotber 2015. [Online]. Available: https://www.torproject.org/about/overview.html.en. [Accessed 2015].

[37] Syverson, Paul, and Griffin Boyce., "Genuine onion: Simple, Fast, Flexible, and Cheap Website Authentication," *arXiv preprint ,* 2015.

[38] Matthew Thomas And Aziz Mohaisen , "Measuring the Leakage of Onion at the Root: A Measurement of Tor's. Onion Pseudo-TLD in the Global Domain Name System," In *Proceedings of the 13th Workshop on Privacy in the Electronic Society. ACM,*, 2014.

[39] Makeuseof.com.[Online].Available:https://makeuseof.com/wp-content/uploads/2014/10/image02.png , Data Accessed :18 May ,2015.

[40] A. Semenov, "Analysis of Services in Tor Network: Finnish Segment," *Proceedings of the 12th European Conference on Information Warfare and Security ECIW Academic Conferences Limited,*2013.

[41] K. Krippendorff, Content Analysis: An Introduction To its Methodology, Sage, 2012.

[42] Ling, Zhen, Junzhou Luo, Kui Wu, Wei Yu, and Xinwen Fu, "TorWard: Discovery of Malicious Traffic Over Tor," In *INFOCOM Proceedings IEEE*, 2014.

[43] Tadesse Anberbir ,Tomio Takakra, "Development of Amharic Text-to-Speech System Using Cepstral Method,", ICT development office, Addis Ababa University, Ethiopia, March, 2009.

[44] Moges Ahmed and Sebsibe H/Mariam, "Named Entity Recognition For Amharic Language," Unpublished Masters Thesis ,Department of Computer Science , Addis Ababa University ,pp. 22-25, 2010.

[45] Saba Amsalu and Sisay Fissaha, "Machine Translation for Amharic: Where we are ?," Unpublished Masters Thesis , Informatics Institute of University of Amsterdam, The Netherlands, 2003.

[46] B. Yimam, “የ አማርኛ ሰዋሰው” (in Amharic), Addis Ababa: Mega Books Plc, 2008.

[47] B. Gamback, F. Olsson, A. A. Argaw, and L. Asker, "Methods for Amharic Part-of-Speech," *In Proceedings of the EACL Workshop on Language Technologies for African,* pp104–111, March 2009.

[48] Yew Choong Chew, Yoshiki Mikami and Robin Lee Nagano, "Language Identification of Web Pages Based on Improved N-gram Algorithm," *IJCSI International Journal of Computer Science Issues,V*ol. 8, No. 1, May 11,2011.

[49] A. Chaabane, "Digging Into Anonymous Traffic: A Deep Analysis of the Tor Anonymizing Network," In *Network and System Security (NSS), 2010 4th International Conference on. IEEE*, 2010.

[50] McCoy D, Bauer K, Grunwald D, Kohno T and Sicker D., "Shining Light In Dark Places: Understanding the Tor Network.," *In Privacy Enhancing Technologies , Springer Berlin Heidelberg.*, pp. 63-76, 2008.

[51] N. Christin, "Traveling the Silk Road: A Measurement Analysis of a Large Anonymous Online Marketplace," In *Proceedings of the 22nd international Conference on World Wide Web. International World Wide Web Conferences Steering Committee*, 2013.

[52] J. Franklin, A. Perrig V. Paxson and S. Savage., "An Inquiry Into the Nature and Causes of the Wealth of Internet Miscreants," *In ACM Conference on Computer and Communications Security*, October,2007.

[53] Michael Chertoff and Tobby Simon, "The Impact of the Dark Web on Internet Governance and Cyber Security," *Centre for International Governance Innovation and, the Royal Institute for International Affairs*, Paper Series No. 6 ,Feb,2015.

[54] L'huillier, Gastón, Sebastián A. Ríos, Hector Alvarez, and Felipe Aguilera., "Topic-Based Social Network Analysis for Virtual Communities of Interests in the Dark Web," *ACM SIGKDD Workshop on Intelligence and Security Informatics. ACM*, 2010.

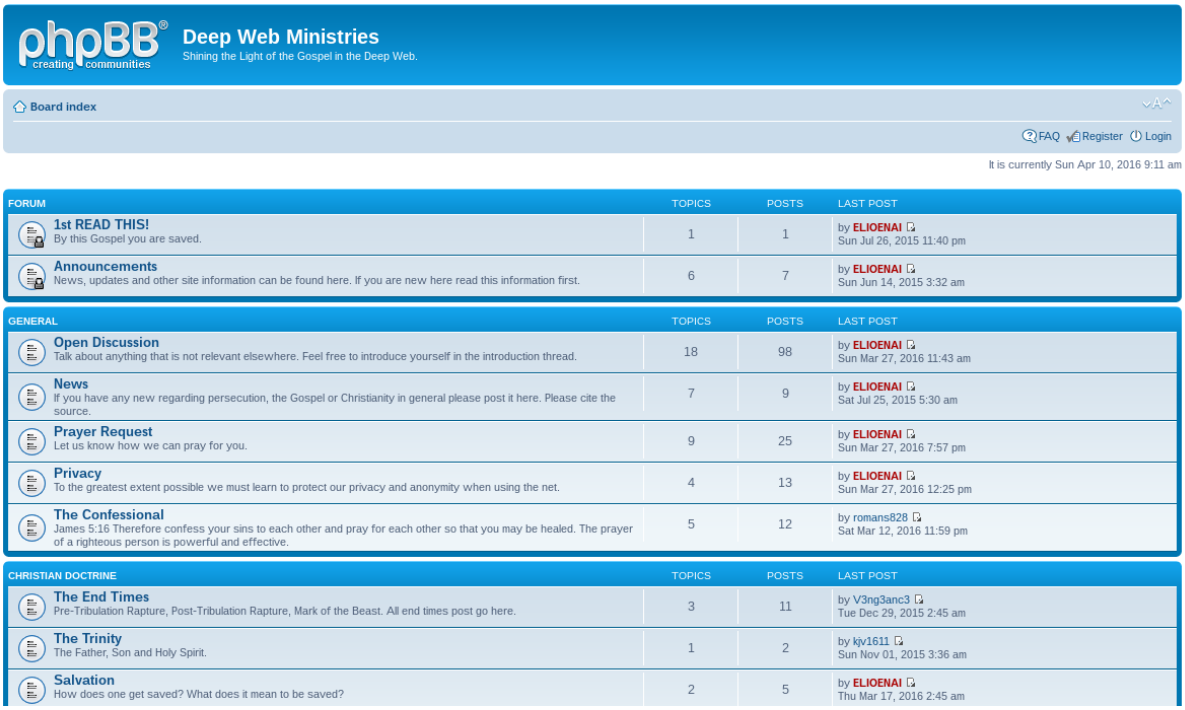
[55] Jennifer Xu , Hsinchun Chen , Yilu Zhou , and Jialun Qin "On the Topology of the Dark Web of Terrorist Groups," *In Intelligence and Security Informatics*, Springer Berlin Heidelberg, 2006.

[56] Ríos, Sebastián A., and Ricardo Muñoz, "Dark Web Portal Overlapping Community Detection Based on Topic Models," *In Proceedings of the ACM SIGKDD Workshop on Intelligence and Security Informatics*, pp.2,2012.

[57] TheTorProject, "TorProject," TheTorProject, 2016. [Online]. Available: https://metrics.torproject.org/userstats-relay-country.html. [Accessed 29 Feb 2016].

[58] D. Brown, "Resilient Botnet Command and Control With Tor," in *DEF CON 18* , 2010.

Annex 1: Dark Web Gospel Ministries Website



phpBB creating communities

Deep Web Ministries

Shining the Light of the Gospel in the Deep Web.

Board index

FAQ Register Login

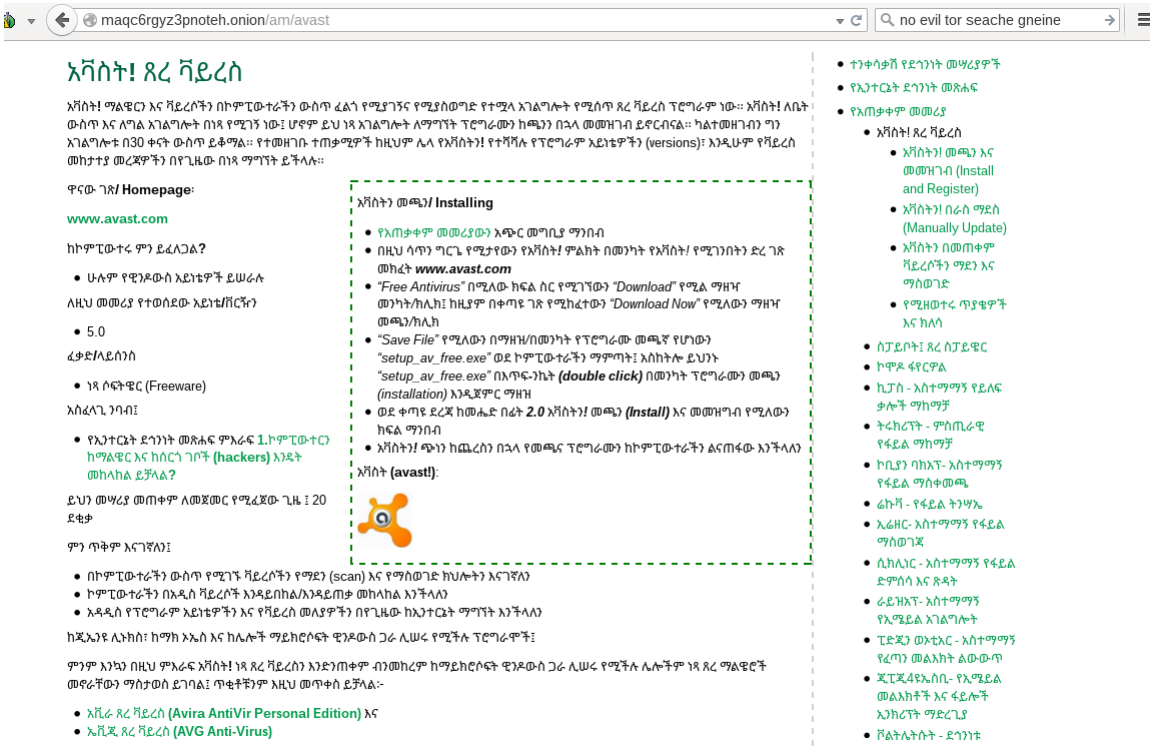
It is currently Sun Apr 10, 2016 9:11 am

FORUM	TOPICS	POSTS	LAST POST
1st READ THIS! By this Gospel you are saved.	1	1	by ELIOENAI Sun Jul 26, 2015 11:40 pm
Announcements News, updates and other site information can be found here. If you are new here read this information first.	6	7	by ELIOENAI Sun Jun 14, 2015 3:32 am

GENERAL	TOPICS	POSTS	LAST POST
Open Discussion Talk about anything that is not relevant elsewhere. Feel free to introduce yourself in the introduction thread.	18	98	by ELIOENAI Sun Mar 27, 2016 11:43 am
News If you have any new regarding persecution, the Gospel or Christianity in general please post it here. Please cite the source.	7	9	by ELIOENAI Sat Jul 25, 2015 5:30 am
Prayer Request Let us know how we can pray for you.	9	25	by ELIOENAI Sun Mar 27, 2016 7:57 pm
Privacy To the greatest extent possible we must learn to protect our privacy and anonymity when using the net.	4	13	by ELIOENAI Sun Mar 27, 2016 12:25 pm
The Confessional James 5:16 Therefore confess your sins to each other and pray for each other so that you may be healed. The prayer of a righteous person is powerful and effective.	5	12	by romans828 Sat Mar 12, 2016 11:59 pm

CHRISTIAN DOCTRINE	TOPICS	POSTS	LAST POST
The End Times Pre-Tribulation Rapture, Post-Tribulation Rapture, Mark of the Beast. All end times post go here.	3	11	by V3ng3anc3 Tue Dec 29, 2015 2:45 am
The Trinity The Father, Son and Holy Spirit.	1	2	by kjv1611 Sun Nov 01, 2015 3:36 am
Salvation How does one get saved? What does it mean to be saved?	2	5	by ELIOENAI Thu Mar 17, 2016 2:45 am

Annex 2: Dark Web Site in Amharic Language



maqc6rgyz3pnoteh.onion/am/avast

no evil tor seache gneine

አቫስት! ጸረ ቫይረስ

አቫስት! ማልዌርን እና ቫይረሶችን በኮምፒውተራችን ውስጥ ፈልጎ የሚያገኝና የሚያስወግድ የተሟላ አገልግሎት የሚሰጥ ጸረ ቫይረስ ፕሮግራም ነው። አቫስት! ለቤት ውስጥ እና ለግል አገልግሎት በነጻ የሚገኝ ነው፤ ሆኖም ይህ ነጻ አገልግሎት ለማግኘት ፕሮግራሙን ከጫንን በኋላ መመዝገብ ይኖርብናል። ካልተመዘገብን ግን አገልግሎቱ በ30 ቀናት ውስጥ ይቆማል። የተመዘገቡ ተጠቃሚዎች ከዚህም ሌላ የአቫስትን! የተሻሻሉ የፕሮግራም አይነቴዎችን (versions)፣ እንዲሁም የቫይረስ መከታተያ መረጃዎችን በየጊዜው በነጻ ማግኘት ይችላሉ።

ዋናው ገጽ/ **Homepage**፡

www.avast.com

ከኮምፒውተሩ ምን ይፈለጋል?

- ሁሉም የዊንዶውስ አይነቴዎች ይሠራሉ

ለዚህ መመሪያ የተወሰደው አይነቴ/ቨርዥን

- 5.0

ፈቃድ/ላይሰንስ

- ነጻ ሶፍትዌር (Freeware)

አስፈላጊ ንባብ፤

- የኢንተርኔት ደኅንነት መጽሐፍ ምእራፍ 1.ኮምፒውተርን ከማልዌር እና ከሰርጎ ገቦች (hackers) እንዴት መከላከል ይቻላል?

ይህን መሣሪያ መጠቀም ለመጀመር የሚፈጀው ጊዜ ፤ 20 ደቂቃ

ምን ጥቅም እናገኛለን፤

- በኮምፒውተራችን ውስጥ የሚገኙ ቫይረሶችን የማደን (scan) እና የማስወገድ ክህሎትን እናገኛለን
- ኮምፒውተራችን በአዲስ ቫይረሶች እንዳይበከል/እንዳይጠቃ መከላከል እንችላለን
- አዳዲስ የፕሮግራም አይነቴዎችን እና የቫይረስ መለያዎችን በየጊዜው ከኢንተርኔት ማግኘት እንችላለን

ከጂኤንዩ ሊኑክስ፣ ከማክ ኦኤስ እና ከሌሎች ማይክሮሶፍት ዊንዶውስ ጋራ ሊሠሩ የሚችሉ ፕሮግራሞች፤

ምንም እንኳን በዚህ ምእራፍ አቫስት! ነጻ ጸረ ቫይረስን እንድንጠቀም ብንመክረም ከማይክሮሶፍት ዊንዶውስ ጋራ ሊሠሩ የሚችሉ ሌሎችም ነጻ ጸረ ማልዌሮች መኖራቸውን ማስታወስ ይገባል፤ ጥቂቶቹንም እዚህ መጥቀስ ይቻላል:-

- አቪራ ጸረ ቫይረስ (Avira AntiVir Personal Edition) እና
- ኤቪጂ ጸረ ቫይረስ (AVG Anti-Virus)

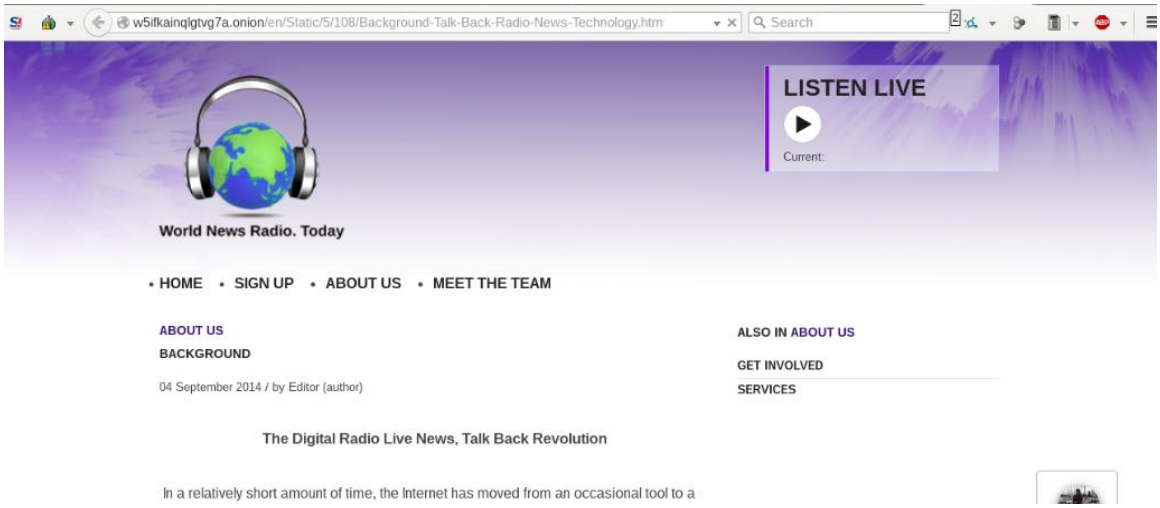
አቫስትን መጫን/ Installing

- የአጠቃቀም መመሪያውን አጭር መግቢያ ማንበብ
- በዚህ ሳጥን ግርጌ የሚታየውን የአቫስት! ምልክት በመንካት የአቫስት! የሚገንበትን ድረ ገጽ መክፈት ***www.avast.com***
- *"Free Antivirus"* በሚለው ክፍል ስር የሚገኘውን *"Download"* የሚል ማዘዣ መንካት/ክሊክ፤ ከዚያም በቀጣዩ ገጽ የሚከፈተውን *"Download Now"* የሚለውን ማዘዣ መጫን/ክሊክ
- *"Save File"* የሚለውን በማዘዝ/በመንካት የፕሮግራሙ መጫኛ የሆነውን *"setup_av_free.exe"* ወደ ኮምፒውተራችን ማምጣት፤ አስከትሎ ይህንኑ *"setup_av_free.exe"* በአጥፍጋኬት ***(double click)*** በመንካት ፕሮግራሙን መጫን *(installation)* እንዲጀምር ማዘዝ
- ወደ ቀጣዩ ደረጃ ከመሔድ በፊት ***2.0*** አቫስትን! መጫን ***(Install)*** እና መመዝግብ የሚለውን ክፍል ማንበብ
- አቫስትን! ጭነን ከጨረስን በኋላ የመጫና ፕሮግራሙን ከኮምፒውተራችን ልናጠፋው እንችላለን

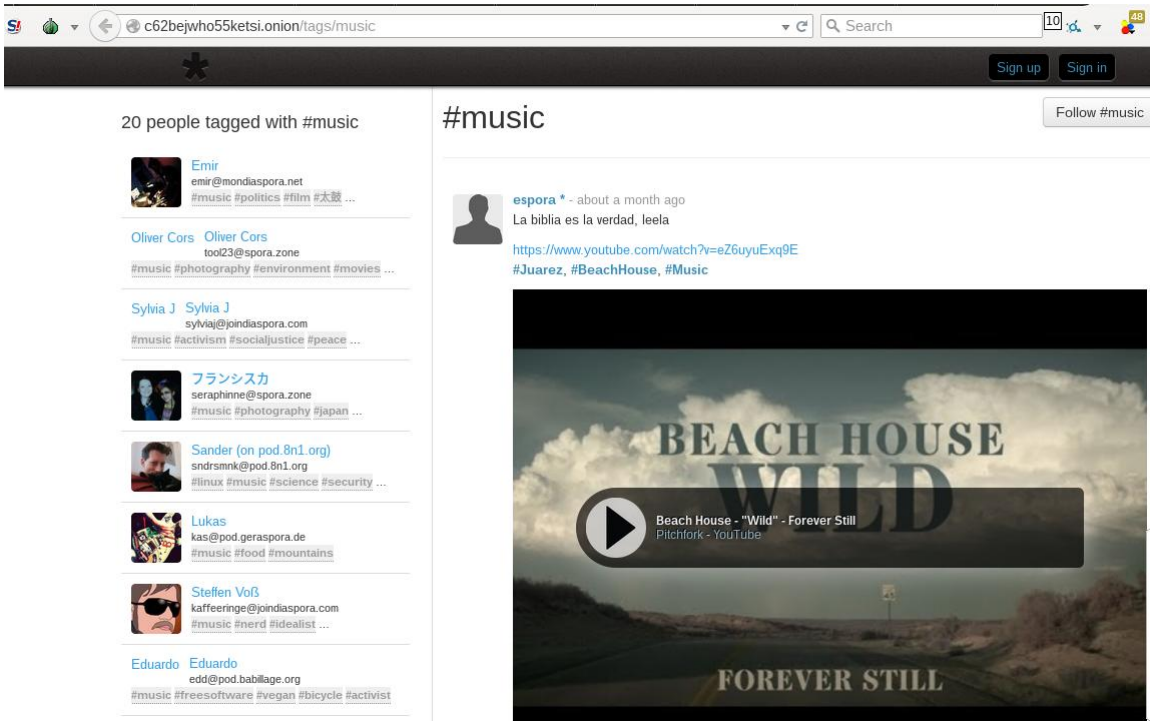
አቫስት **(avast!)**:

- ስፓይቦት፤ ጸረ ስፓይዌር
- ኮሞዶ ፋየርዎል
- ኪፓስ - አስተማማኝ የይለፍ ቃሎች ማከማቻ
- ትሩክሪፕት - ምስጢራዊ የፋይል ማከማቻ
- ኮቢያን ባክአፕ- አስተማማኝ የፋይል ማስቀመጫ
- ሬኩቫ - የፋይል ትንሣኤ
- ኢሬዘር- አስተማማኝ የፋይል ማስወገጃ
- ሲክሊነር - አስተማማኝ የፋይል ድምሰሳ እና ጽዳት
- ራይዝአፕ- አስተማማኝ የኢሜይል አገልግሎት
- ፒድጂን ውኦቲአር - አስተማማኝ የፈጣን መልእክት ልውውጥ
- ጂፒጂ4ዩኤስቢ- የኢሜይል መልእክቶች እና ፋይሎች ኢንክሪፕት ማድረጊያ
- ቮልትሌትሱት - ደኅንነቱ

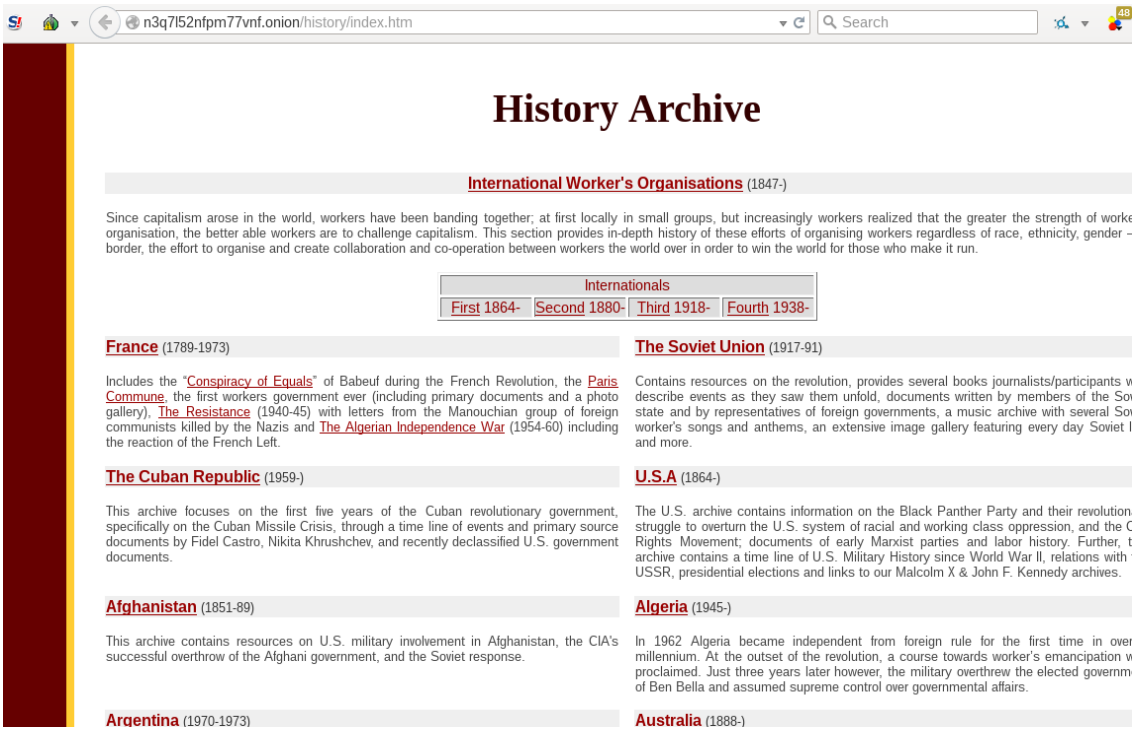
Annex 3: Dark Web Live Radio News Website



Annex 4: Dark Web Social Networking and Music Website



Annex 5: Dark Web World History Archives



I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

Declared by:

Name: **Daniel Adenew Wondyifraw**

Signature: _____

Date: _____

Confirmed by advisor:

Name: **Dr. Mesfin Kifle**

Signature: _____

Date: _____