Czech University of Life Sciences Prague

Faculty of Economics and Management

Department Of Information Technologies

Bachelor Thesis

Design of an Algorithm for Crawling Web Articles focusing on the given Topic

Author: Abdulbasit Ayinde Aliyu

Supervisor:Ing. Jan Masner, Ph.D.

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Abdulbasit Ayinde Aliyu

System Engineering & Informatics

Thesis title

Design of an algorithm for crawling web articles focusing on the given topic

---

Objectives of thesis

Main objective

The main objective of this thesis is to design and implement an algorithm that will crawl web articles focusing on a particular topic. In this thesis, the algorithm will be validated on articles focusing on the agrarian sector.

Partial objectives

- To create a review of similar solutions.
- To select a suitable technology for implementing the algorithm.
- To validate the algorithm in the context of English-language websites.

Methodology

The methodology of solving the theoretical part of the Bachelor Thesis will be based on the study and analysis of professional information sources. Based on the knowledge gained in the theoretical part of the work, the best and most suitable algorithm will be implemented using the most appropriate technologies in the experimental application. Subsequently, appropriate testing will be performed to evaluate the algorithm, focusing on efficiency and accuracy. Based on the synthesis of theoretical knowledge and the results of the practical part, the conclusions will be formulated.

The proposed extent of the thesis

40 – 50 pages

Keywords

Web crawler, Web Crawling Algorithms, Search Engine

Recommended information sources

kumar, Rahul, Jain, Anurag, Agrawal, Chetan; Survey of Web Crawling Algorithms;An International Journal (AVC) Vol.3, No.3, Sep 2016

Matthew Turland. php|architect's Guide to Web Scraping Contents; Marco Tabini & Associates, Inc 2009–2010

Michael Schrenk. Webbots, Spiders, and Screen Scrapers 2nd Edition; No Starch Press, Inc. 2012 Prashant Dahiwale, Rashmi Janbandhu, M. M. Raghuwanshi. Analysis of web crawling algorithms;

Research Gate 2014

Ruthe Grubbs – Web Search Engines and Optimization,Library Press Revised Edition:2016

Expected date of thesis defence

2022/23 SS – FEM

The Bachelor Thesis Supervisor

Ing. Jan Masner, Ph.D.

Supervising department

Department of Information Technologies

Electronic approval: 11. 8. 2022

doc. Ing. Jiří Vaněk, Ph.D.

Head of department

Electronic approval: 27. 10. 2022 **doc. Ing. Tomáš Šubrt, Ph.D.**

Dean

Prague on 13. 02. 2023

Declaration

I declare that I have worked on my bachelor thesis titled "Design of an Algorithm for Crawling Web Articles focusing on the given Topic" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break any copyrights.

In Prague on 15th of March 2023 _____

Acknowledgement

I would like to thank my supervisor, Ing. Jan Masner, Ph. D., for your time, instructions, and advice, which were very helpful and essential during the writing of this thesis. I would also like to thank Mr. and Mrs. Igbasan for their support and assistance throughout the course of my thesis.

Design of an Algorithm for Crawling Web Articles focusing on the given Topic

Abstract

The purpose of this thesis is to demonstrate and generate a review of several crawling algorithms; also, the thesis's sub-goal is to build and implement an algorithm that crawls online pages.

The theoretical part of this thesis provides an overview of the history of the internet, networks, and the connection between the web and the internet. A detailed analysis of the World Wide Web and Markup Language, which is used as the template display engine for the WWW, is presented. The concept of hyperlinks is explored, and different types of web crawling algorithms are studied to determine their strengths and weaknesses. Based on the evaluation, a suitable algorithm is chosen for implementation.

The practical part of the thesis involves the implementation of the selected algorithm, and the most suitable technology is chosen for implementation. The results are then compared based on response time and relevance. Through this comparison, the Focused Algorithm is found to be the most suitable algorithm due to its quick response time.

Overall, this thesis provides valuable insights into the design and implementation of an algorithm for crawling web articles focusing on a given topic. The resulting algorithm provides a useful tool for businesses and researchers to gather insights and data from online sources, making it a valuable resource.

Keywords: Web, Webbot,Algorithm, Crawler, Internet, www, Web Crawling, HTML

Návrh algoritmu pro procházení webových článků se zaměřením na dané téma

Abstrakt

Účelem této práce je demonstrovat a vytvořit přehled několika algoritmů procházení; Dílčím cílem práce je také sestavit a implementovat algoritmus, který prochází online stránky.

Teoretická část této práce poskytuje přehled historie internetu, sítí a propojení mezi webem a internetem. Je prezentována podrobná analýza World Wide Web a Markup Language, který se používá jako šablonový zobrazovací engine pro WWW. Je zkoumán koncept hypertextových odkazů a studovány různé typy algoritmů procházení webu, aby se určily jejich silné a slabé stránky. Na základě vyhodnocení je vybrán vhodný algoritmus pro implementaci.

Praktická část práce zahrnuje implementaci zvoleného algoritmu a pro implementaci je vybrána nejvhodnější technologie. Výsledky jsou poté porovnány na základě doby odezvy a relevance. Prostřednictvím tohoto srovnání bylo zjištěno, že Focused Algorithm je nejvhodnějším algoritmem díky své rychlé době odezvy.

Celkově tato práce poskytuje cenné poznatky o návrhu a implementaci algoritmu pro procházení webových článků zaměřených na dané téma. Výsledný algoritmus poskytuje podnikům a výzkumníkům užitečný nástroj ke shromažďování poznatků a dat z online zdrojů, což z něj činí cenný zdroj.

Klíčová slova: Web,Webbot,Algoritmus,Prohledávač,Internet,www, Procházení webu,HTML

Table of content

BACHELOR THESIS ASSIGNMENT **2**

Design of an algorithm for crawling web articles focusing on the given topic **2**

Methodology **2**

The proposed extent of the thesis **3**

Keywords **3**

Recommended information sources **3**

Expected date of thesis defence **3**

The Bachelor Thesis Supervisor **3**

Supervising department **3**

doc. Ing. Jiří Vaněk, Ph.D **3**

1 Introduction **10**

2 Objectives and Methodology **11**

2.1 Objectives 11

2.2 Methodology 11

3 Literature Review **12**

3.1 Internet 12

3.1.1 Primary Network 12

3.1.2 Interconnect of Networks 14

3.1.3 Computer Location 16

3.1.4 Web & Internet 17

3.1.5 Extranet & Internet 17

3.2 World Wide Web 18

3.3 HTML5 19

3.3.1 HTML Semantic Elements 20

3.3.2 HTML <section> Elements 21

3.3.3 HTML <article> Elements 21

3.3.4 HTML <nav> Elements 22

3.3.5 HTML <aside> Elements 23

3.3.6 HTML <figure> and <figcaption> Elements 23

3.4 HTML Hyperlinks 24

3.4.1 Internal Hyperlink 24

3.4.2 External Hyperlink 25

3.4.3 HTML Websites & Search Engines 25

3.5 URL....26
3.6 Web Crawling....27
3.6.1 Focused Crawler....27
3.6.2 Incremental Crawler....28
3.6.3 Continuous Crawler....28
3.6.4 Parallel Crawler....28
3.6.5 Distributed Crawler....28
3.7 Web Crawling Algorithm....28
3.7.1 Breadth First Search (BFS)....29
3.7.2 Depth First Search....29
3.7.3 Page Rank....30
3.7.4 Path-Ascending Algorithm....30
3.7.5 Genetic Algorithm....31
3.8 A Comparative Analysis of Web Crawlers and Crawling Algorithms....31
3.9 Web Crawling Algorithm....33
3.10 Summary....33
3.11 Challenges....34

4 Practical Part....35
4.1 Research Design....35
4.2 Data Collection....35
4.3 Algorithm Design....36
4.4 Implementation....36
4.5 AgrarianSpider Web Crawling....37

5 Results and Discussion....45
5.1 Results....45
5.2 Discussion....47

6 Conclusion....48

7 References....49

8 List of pictures, tables, graphs and abbreviations....51
8.1 List of pictures....51
8.2 List of tables....51
8.3 List of Source Code....52

1 Introduction

The World Wide Web, or simply the Web, has grown exponentially in recent years to become the largest repository of information ever created and has become a common and habitual resource for a variety of entities, from individuals to entire organizations. On general web pages, web blogs, web forums, e-commerce websites, and social networking websites, a growing number of entities are registering large quantities of facts and events, news, formal and informal knowledge, people's opinions, and social interactions, thereby creating a vast and valuable public information repository. Therefore, the Internet is a fresh and current source of potentially useful information.

But because the Web is a decentralized, highly dynamic environment comprised of noisy, highly unstructured, heterogeneous data, and filled with a vast and growing volume of information, locating, and retrieving its relevant data in a large-scale automated manner for use and content analysis are unquestionably difficult tasks. These facts present both opportunities and challenges for the study of techniques and strategies for searching, retrieving, and evaluating web-based data.

The primary objective of this thesis is to demonstrate and evaluate various crawling algorithms; the secondary objective is to develop and implement a crawling algorithm for web pages.

2 Objectives and Methodology

2.1 Objectives

The main objective of this thesis is to design and implement an algorithm that will crawl web articles focusing on a particular topic. In this thesis, the algorithm will be validated on articles focusing on the agrarian sector.
The partial objectives of this thesis are:

- To create a review of similar solutions.
- To select a suitable technology for implementing the algorithm.
- To validate the algorithm in the context of English-language websites.

2.2 Methodology

The methodology of solving the theoretical part of the Bachelor Thesis will be based on the study and analysis of professional information sources. Based on the knowledge gained in the theoretical part of the work, the best and most suitable algorithm will be implemented using the most appropriate technology in the practical part into the experimental complete application, subsequently, appropriate unit testing will be performed and evaluated, focusing on the efficiency of the selected algorithm. Based on the synthesis of theoretical knowledge and results of the practical part, the conclusions of the work will be formulated.

3 Literature Review

3.1 Internet

The Internet has brought about a significant transformation in the computer and communications industries. The development of various technologies such as the telegraph, telephone, radio, and computer made it possible to combine these capabilities and create a global platform for broadcasting, information transmission, and collaboration. Continued investment and commitment to information infrastructure research and development have been crucial to the Internet's success, and it serves as a prime example of the benefits of such initiatives. Today, even the average person on the street is familiar with terms such as "http://www.acm.org" and "bleiner@computer.org" [1].

The National (or Global or Galactic) Information Infrastructure is considered the earliest prototype of the Internet. Its complex history involves several organizational, technological, and social factors. As online technologies continue to gain importance in areas such as electronic commerce, information collection, and community activities, the Internet's influence extends beyond computer communications and permeates all aspects of society [1].

The Internet serves as the Web's technical foundation or supporting infrastructure. Essentially, it is a vast network of communicating computers that are interconnected. Its origins can be traced back to a research project funded by the U.S. military in the 1960s. Since then, public universities and private companies have played a significant role in its evolution into a public infrastructure in the 1980s. While the underlying technologies have continued to evolve, the Internet's operation has remained relatively unchanged, serving as a mechanism for connecting computers and ensuring that they remain connected [2].

The Internet has transformed the computer and communications industries, and its success serves as a testament to the benefits of continued investment and commitment to information infrastructure research and development. As online technologies continue to grow in importance, the Internet's influence extends beyond computer communications and permeates all aspects of society, making it a crucial part of our daily lives.

3.1.1 Primary Network

The primary network is a crucial aspect of computer communication that enables two or more computers to communicate with each other. This connection can be established either physically, such as through an Ethernet cable, or wirelessly, such as through Wi-Fi or Bluetooth systems. In modern times, most computers come equipped with the necessary hardware and software to establish these connections with ease [2].

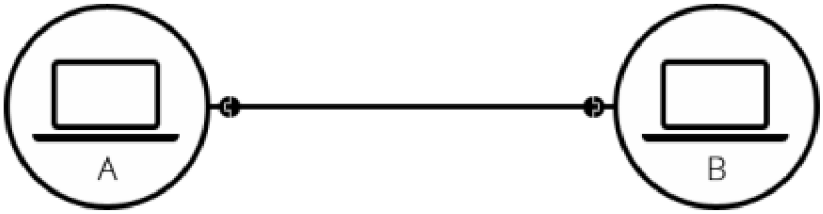

Figure 1. simple connection between two computers. Source:[2]

Figure 1 provides an illustration of a simple connection between two computers, which serves as the foundation of a primary network. However, when multiple computers need to be connected, the situation becomes more complicated. For instance, connecting ten computers would require 45 cables, with nine plugs per computer. This situation is both unmanageable and impractical [2].

To address this issue, each computer on a network is connected to a miniature computer, known as a router. The router plays the role of a signaller at a train station, ensuring that messages are sent to the correct destination computer, as shown in Figure 2.

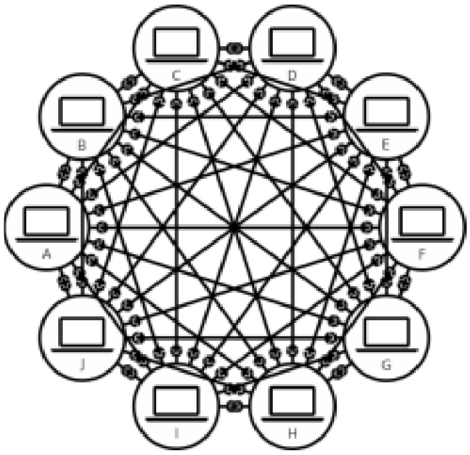

Figure 2. Computer network connection between more computers. Source:[2]

To send a message from computer A to computer B, computer A sends the message to the router, which then ensures that the message does not reach computer C. This process ensures that messages are delivered efficiently and effectively, reducing errors and increasing the speed of communication.

With the addition of a router to the system, the number of cables needed to connect ten computers reduces significantly. Instead of requiring 45 cables, only ten cables are needed, one for each computer and a router with ten plugs, as shown in Figure 3. This significant reduction in cables makes it more manageable to connect multiple computers on a network and ensures that the network runs efficiently and effectively [2].

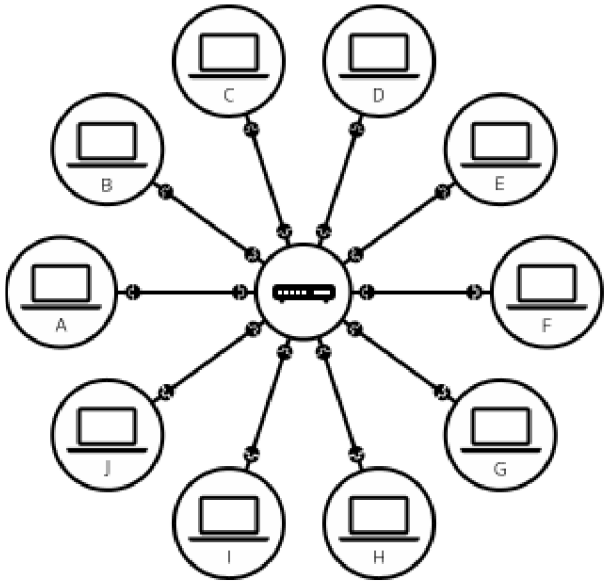

Figure 3. Router with cabled-connected computers. Source:[2]

The primary network is the foundation of computer communication, enabling computers to connect with each other either physically or wirelessly. The addition of a router to the network ensures that messages are sent to the correct destination computer, reducing errors, and increasing the speed of communication. With the use of routers, it becomes easier to connect multiple computers on a network, significantly reducing the number of cables required, making the network more manageable and efficient [2].

3.1.2 Interconnect of Networks

Connecting tens of millions, if not billions, of computers requires a more complex network infrastructure than a simple router-to-computer connection. While routers serve as computers, connecting a single router cannot scale to this level. However, connecting two routers together using a cable expands the network's capabilities indefinitely [2].

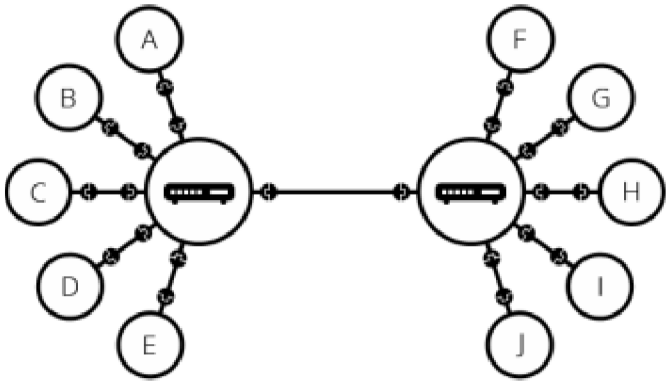

Figure 4. Two Routers connection with a cable. Source:[2]

Figure 4 illustrates two routers connected with a cable, but we can connect multiple routers together to form a larger network, as shown in Figure 5.

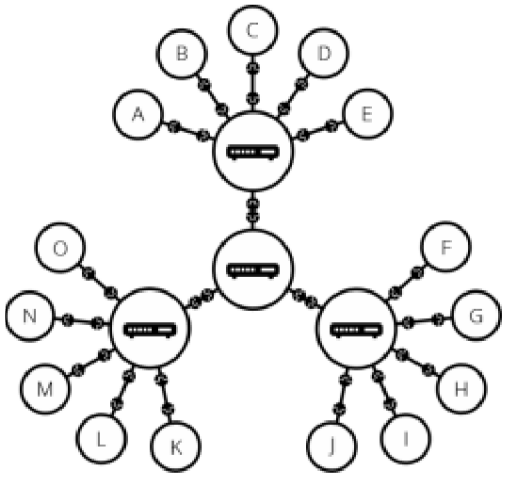

Figure 5. Four Routers connection via cable. Source:[2]

The network created through the connection of routers is like the internet, but something is missing. Everyone may have their own computer network and connecting cables between each of these networks and the rest of the world is impractical. However, using the existing telephone infrastructure can help address this issue. The telephone infrastructure already connects homes to anyone in the world, and a modem is a specialized piece of hardware that can connect our network to the telephone network [2].

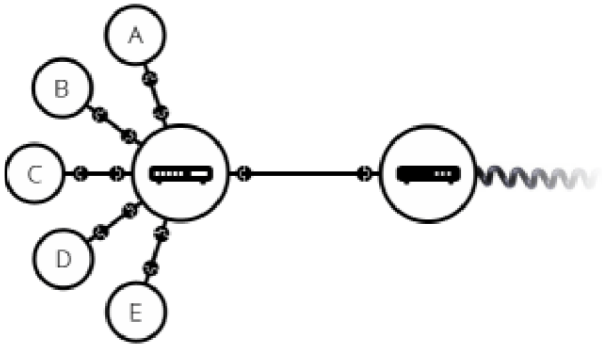

Figure 6. Router connection with a modem. Source:[2]

Figure 6 depicts a connected router and modem, with the modem converting data from our network into data that the telephone infrastructure can handle and vice versa.

Once connected to the telephone infrastructure, the next step is to send messages from our network to the network with which we wish to communicate. To achieve this, we connect our network to an Internet Service Provider (ISP), a company that manages a network of special routers that are interconnected and can access the routers of other ISPs. The message from our network is then routed via the network of ISP networks to the destination network [2].

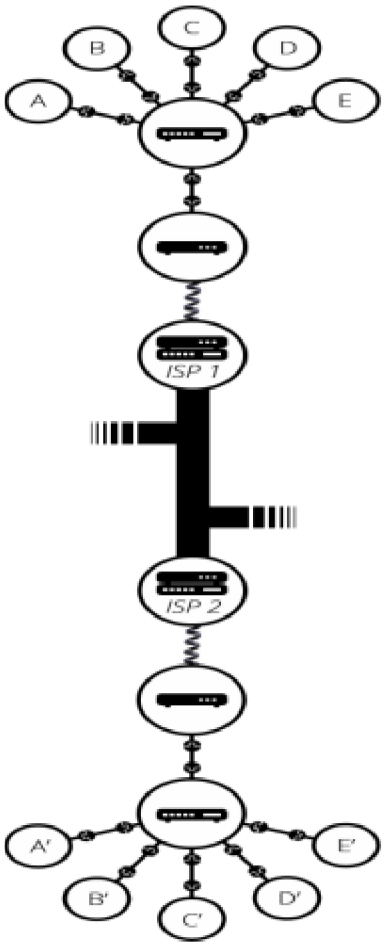

Figure 7. Connection between ISP and Final user. Source:[2]

Figure 7 illustrates the connection between the ISP and the final user. The internet is entirely made up of this network infrastructure, enabling the efficient and effective transmission of data across the globe.

Connecting tens of millions, if not billions, of computers requires a more complex network infrastructure than a simple router-to-computer connection. Connecting multiple routers together using a cable expands the network's capabilities indefinitely. By using the existing telephone infrastructure and connecting to an ISP, the network can efficiently and effectively transmit data across the globe. The internet is entirely made up of this network infrastructure, providing a reliable means of communication between individuals and organizations worldwide [2].

3.1.3 Computer Location

When sending a message, it is essential to specify the destination computer. Every computer connected to a network has a unique identifying address, known as an IP address. An IP address is a four-number address separated by periods, such as 192.168.2.10. However, human beings have difficulty remembering such addresses, so an IP address can be aliased with a human-readable name known as a domain name, making it more convenient. At the time of writing, google.com had the IP address of 142.250.190.78 [2].

Connecting to a computer via the internet is most convenient when using a domain name. Figure 8 illustrates how to locate a computer using its domain name and IP address. When an individual enters a domain name in their browser's address bar, the browser sends a request to a Domain Name Server (DNS). The DNS then converts the domain name into an IP address, which is then used to locate the computer. This process allows individuals to access websites and other online resources without having to remember their IP addresses.

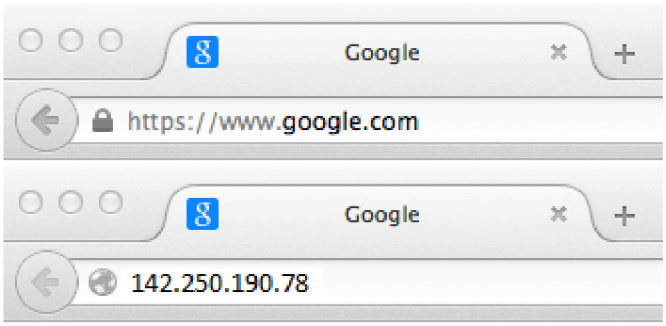

Figure 8. Locate a Computer using its domain name and IP address. Source:[2]

Moreover, every computer connected to a network has a unique identifying address known as an IP address. However, humans have difficulty remembering such addresses, so an IP address can be aliased with a human-readable name known as a domain name, making it more convenient. When connecting to a computer via the internet, using a domain name is most convenient. This process involves converting the domain name into an IP address, which is then used to locate the computer. This process enables individuals to access websites and other online resources without having to remember their IP addresses, making the internet more accessible and user-friendly.

3.1.4 Web & Internet

The Internet and World Wide Web are often used interchangeably, but they are not the same thing. The Internet is a technical infrastructure that links billions of computers together, whereas the World Wide Web is a service built upon it. The Internet is a network of networks that allows computers to communicate and share data, while the World Wide Web is a vast network of interconnected documents and resources accessed through a web browser. Other services, including email and IRC, are also built on top of the Internet. This distinction is important to understand as it highlights the various services and applications that are possible through the infrastructure provided by the Internet [2].

3.1.5 Extranet & Internet

Intranets and extranets are private networks that utilise the same infrastructure and protocols as the Internet. Intranets are accessible only to members of a particular organisation, providing a secure portal for members to collaborate, communicate, and access shared resources. For instance, an intranet may host web pages for sharing department or team information, shared drives for managing important documents and files, portals for performing business administration tasks, and collaboration tools such as wikis, message boards, and instant messaging systems. Extranets, on the other hand, allow the sharing and collaboration of a private network with other organisations, typically used

to securely share information with clients and stakeholders who work closely with a business. Their functions frequently resemble those of an intranet: file and information sharing, collaboration tools, message boards, etc. [2].

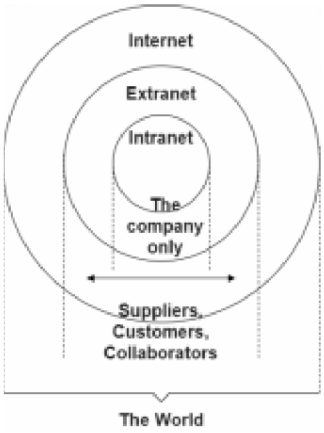

Figure 9. simple formation of the Internet and the World. Source:[2]

The distinction between the Internet and World Wide Web is essential to understand, as it highlights the various services and applications that are possible through the infrastructure provided by the Internet. Intranets and extranets utilize the same infrastructure and protocols as the Internet, providing secure portals for members to collaborate, communicate, and access shared resources within their organization or with other organizations. These private networks are essential for businesses to operate efficiently and securely while still utilizing the infrastructure and protocols provided by the Internet. Figure 9 illustrates the formation of the Internet and the world, highlighting the vast network of interconnected devices that make up the Internet and the various applications and services that run on top of it.

3.2 World Wide Web

The World Wide Web (WWW) is a global network of publicly accessible web pages that can be accessed via the Internet. The Web is one of many applications built on top of the Internet; it is distinct from the Internet. The architecture of the World Wide Web was proposed by Tim Berners-Lee in 1990, when he invented HTTP, HTML, and URL on his computer at the CERN physics research laboratory. He announced his creation in 1991, marking the first public appearance of the World Wide Web. The system currently known as "the Web" consists of several components, including the HTTP protocol, unique universal identifiers known as URLs, and hypertext markup language (HTML) used for publishing web documents. Additionally, linking resources via hyperlinks is a defining concept of the World Wide Web that contributes to its identity as a collection of interconnected documents [3].

After Tim Berners-Lee invented the World Wide Web, he founded the World Wide Web Consortium (W3C) to standardize and further develop it. This consortium consists of important Web interest groups, such as web browser developers, government entities, researchers, and universities, and is tasked with outreach and education. The functioning of the Internet, which is the infrastructure on top of which the World Wide Web operates, involves both clients and servers that are connected to the Internet. Clients are the internet-connected devices of the average web user, such as a computer connected to Wi-Fi or a

mobile phone connected to a mobile network, and the web-accessing software available on those devices. Web pages, websites, and applications are stored on servers, and when a client device requests a web page, a copy is downloaded from the server and displayed in the user's web browser. In addition to clients and servers, the Internet also involves internet connections, communication protocols that define internet data transmission, the Domain Name System (DNS), and HTTP [4].

When a user types a web address into their browser, the browser sends an HTTP request message to the server requesting that it sends a copy of the website to the client, and this message, along with all other data sent between the client and server, is transmitted over the Internet connection using TCP/IP. If the server accepts the client's request, it sends the client an email containing the message "200 OK," which translates to "Of course you can view that website!" and then begins sending the website's files to the browser as a series of small chunks called data packets. The browser assembles the fragments into an entire web page and displays it to the user. In addition, a DNS server is consulted to determine the IP address of the website before retrieving it, and HTML files frequently contain elements that refer to external CSS stylesheets and external JavaScript scripts, which are parsed in a specific order [4].

The World Wide Web is a service built upon the infrastructure of the Internet, and it consists of several components, including the HTTP protocol, unique universal identifiers known as URLs, and hypertext markup language (HTML) used for publishing web documents. The functioning of the Internet involves both clients and servers, internet connections, communication protocols that define internet data transmission, the Domain Name System (DNS), and HTTP. When a user types a web address into their browser, the browser sends an HTTP request message to the server requesting that it sends a copy of the website to the client, and this message is transmitted over the Internet connection using TCP/IP.

3.3 HTML5

HTML (Hypertext Markup Language) is a widely used markup language that defines the structure and content of web pages on the World Wide Web. HTML was initially designed for the creation of scientific documents with the aim of enabling document semantic description. However, its versatility has allowed it to evolve over the years and be used for a wide range of document types and even applications. HTML is an essential part of the web infrastructure and is used by developers to create everything from static web pages to complex web applications [5].

In its early years (1990-1995), HTML underwent several revisions and extensions primarily led by CERN and the IETF. The development of HTML shifted to the World Wide Web Consortium (W3C) in the late 1990s, with HTML 3.2 and HTML 4 being released in quick succession. The development of XHTML (eXtensible HyperText Markup Language), an XML-based alternative to HTML, began in 2000, which led to the release of XHTML 1.0. XHTML 1.0 was essentially a reformulation of HTML4 in XML format that added no new features other than serialization. However, the development of XHTML2, which was incompatible with HTML and XHTML, ran parallel to XHTML [6].

The development of HTML was rekindled in 2003 following the release of XForms, which sparked renewed interest in evolving HTML itself rather than finding replacements. A proof of concept was created to demonstrate that HTML4's forms could be extended to provide many of the features introduced by XForms 1.0 without requiring browsers to implement rendering engines that were incompatible with existing HTML web

pages. This early work eventually led to the creation of HTML5, which included features such as multimedia support, new elements, and form controls, among others. Apple, Mozilla, and Opera led the development of HTML5 through the WHATWG (Web Hypertext Application Technology Working Group), which jointly owned the specification. In 2015, HTML5 was subsumed under a new HTML specification that included CSS and DOM concepts already in use by W3C's document markup languages such as HTML 4 and XHTML 1 [6].

HTML5 is an essential component of modern web development and offers developers greater flexibility, better multimedia support, and easier integration with other web technologies. Its development was a collaborative effort between the WHATWG and the W3C, with the goal of creating a single HTML version moving forward. The evolution of HTML will continue to meet the needs of web developers and users in the years to come.

3.3.1 HTML Semantic Elements

HTML Semantic Elements are specific tags that have a meaning beyond their presentation. In other words, they provide contextual information about the content they contain. For instance, if we use the <header> tag, we know that the content enclosed within it represents the header section of the webpage. Similarly, if we use the <footer> tag, we know that it represents the footer section of the webpage. This helps developers to write more accessible and meaningful code and assists search engines to better understand the structure of the webpage [7].

Some HTML tags, such as and , are not semantic because they don't provide any context or meaning about the content enclosed within them. These tags can be used to style or structure the content, but they do not provide any information about what the content represents. On the other hand, semantic elements such as <article>, <aside>, <nav>, <header>, and <footer> provide meaningful information to both the browser and developer as in figure 10.

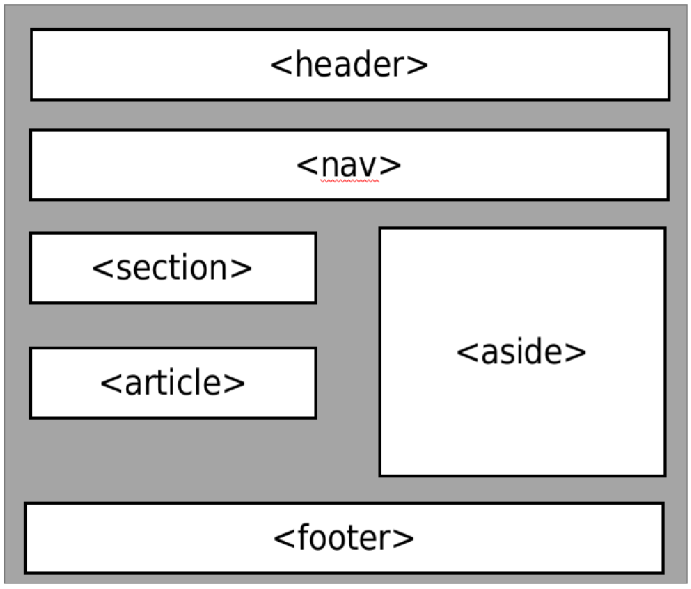

Figure 10. simple HTML Page. Source:[7]

Using semantic elements in HTML is essential for improving the accessibility of web pages. Screen readers, for example, can use the semantic information provided by these tags to provide a better experience for users who rely on them. Additionally, using

semantic elements makes it easier to understand the purpose and structure of a webpage, which is beneficial for developers who are collaborating on a project or maintaining an existing one.

In HTML, there are several semantic elements available to define different parts of a webpage. These include <article> for standalone content, <aside> for supporting content, <details> for additional details, <figcaption> for captions, <figure> for illustrations, <footer> for the footer section, <header> for the header section, <main> for the primary content, <mark> for highlighted text, <nav> for navigation links, <section> for sections of content, <summary> for summary content, and <time> for dates and times. These tags can be used to describe the content enclosed within them, providing additional meaning to the webpage and enhancing its accessibility [7].

3.3.2 HTML <section> Elements

The <section> element in HTML is used to define a section in a document, which is a thematic grouping of content that is often accompanied by a heading. This semantic element allows developers to structure their web pages in a more organized and meaningful way. For instance, a blog post may have multiple sections such as "Introduction," "Body," and "Conclusion," each with its own heading. The <section> element can also be used for other types of content such as chapters, news items, or contact information.

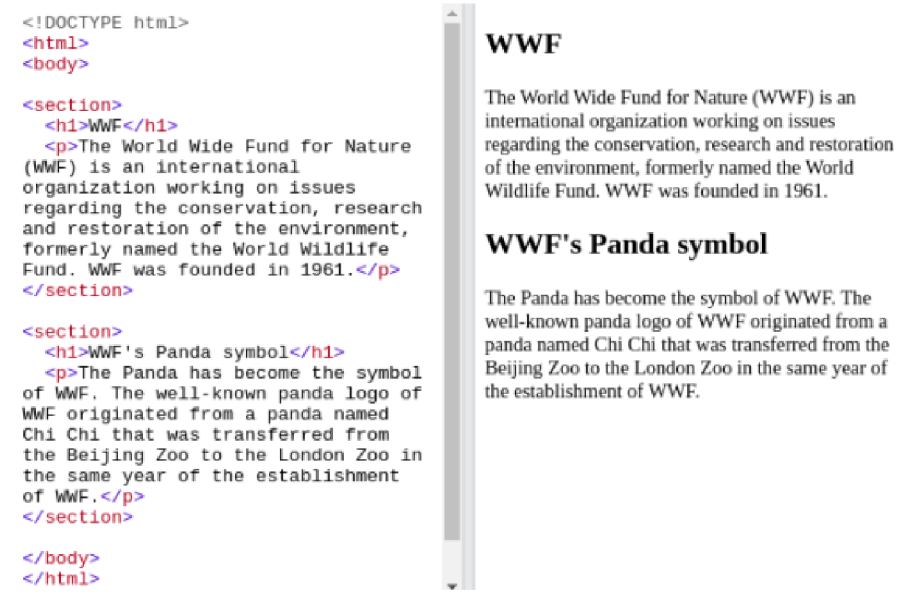

Figure 11. HTML <section> Element. Source:[author]

By using <section> elements, web developers can improve the accessibility of their websites for users with disabilities, such as those using screen readers. Semantic elements like <section> provide more context and help screen readers to better understand the organization of the page. Moreover, search engines like Google use semantic markup to better understand the content of a page and improve the search results.

3.3.3 HTML <article> Elements

The <article> element in HTML is used to define a self-contained and independent piece of content that can be distributed on its own. This means that the content inside an

<article> element should make sense on its own and not depend on the rest of the web page for context. Examples of where the <article> element can be used include blog posts, news articles, forum posts, or user comments [7].

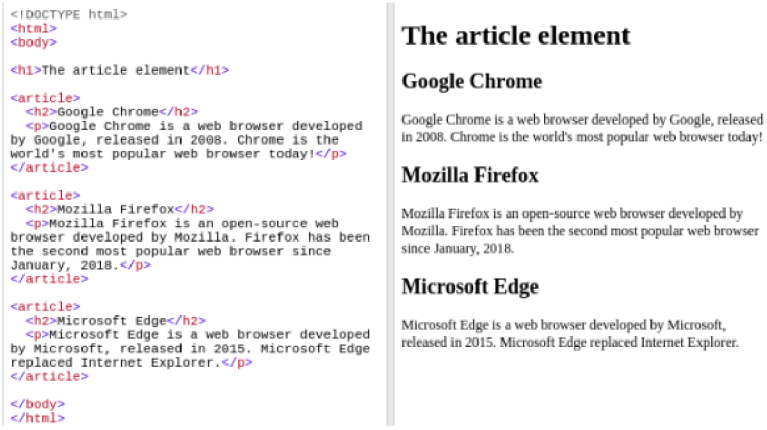

Figure 12. HTML <article> Element. Source:[author]

Using the <article> element helps search engines to better understand the main content of a web page and can improve the search results. Additionally, the use of semantic markup can benefit users with disabilities by providing clearer context and improving the accessibility of the web page.

3.3.4 HTML <nav> Elements

The <nav> element is used to define a set of navigation links in HTML documents. It is an important element for organizing content on web pages, especially for larger websites with a lot of pages. However, it is important to note that not all links in a document should be placed inside a <nav> element. The <nav> element is intended only for major blocks of navigation links [7].

Figure 13. HTML <nav> Element. Source:[author]

The purpose of the <nav> element is to provide a way for browsers, such as screen readers for disabled users, to determine whether to omit the initial rendering of this content. This is because screen readers usually skip over repeated content like navigation links, as this content is not essential to understanding the main content of the page. The <nav> element can also be used to group related links together, such as links to different sections of the same page or links to related pages on the same website.

3.3.5 HTML <aside> Elements

The <aside> element in HTML is used to define some content that is tangentially related to the content it is placed in. Typically, this element is used to define a sidebar, which contains additional information or links that are indirectly related to the main content of the page. The <aside> element can also be used to define a pull-quote or to insert additional information that is not essential to the main content of the page [7].

It is important to note that the content placed inside the <aside> element should be indirectly related to the surrounding content. This means that it should not contain essential information that would be missed if it were removed. Instead, it should contain additional information that would enhance the reader's understanding of the surrounding content.

```
<!DOCTYPE html>
<html>
<head>
<style>
aside {
  width: 30%;
  padding-left: 15px;
  margin-left: 15px;
  float: right;
  font-style: italic;
  background-color: lightgray;
}
</style>
</head>
<body>

<p>My family and I visited The Epcot center this
summer. The weather was nice, and Epcot was amazing!
I had a great summer together with my family!</p>

<aside>
<p>The Epcot center is a theme park at Walt Disney
World Resort featuring exciting attractions,
international pavilions, award-winning fireworks and
seasonal special events.</p>
</aside>

<p>My family and I visited The Epcot center this
summer. The weather was nice, and Epcot was amazing!
I had a great summer together with my family!</p>
<p>My family and I visited The Epcot center this
summer. The weather was nice, and Epcot was amazing!
I had a great summer together with my family!</p>

</body>
```

My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing! I had a great summer together with my family!

My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing! I had a great summer together with my family!

The Epcot center is a theme park at Walt Disney World Resort featuring exciting attractions, international pavilions, award-winning fireworks and seasonal special events.

My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing! I had a great summer together with my family!

Figure 14. HTML <aside> Element. Source:[author]

3.3.6 HTML <figure> and <figcaption> Elements

The <figure> element in HTML is used to define self-contained content, such as illustrations, diagrams, photos, code listings, and more. This element is used to group together content that is related to each other, and can also be used to add captions to the content using the <figcaption> element [7].

The <figcaption> element is used to define a caption for a <figure> element. This element can be placed either as the first or last child of the <figure> element, and is typically used to provide a description or context for the content inside the <figure> element. It is important to note that the <figcaption> element should be used sparingly, and only when a caption is necessary to enhance the reader's understanding of the content [23].

```
<!DOCTYPE html>
<html>
<body>

<h2>Places to Visit</h2>

<p>Puglia's most famous sight is
the unique conical houses (Trulli)
found in the area around
Alberobello, a declared UNESCO
World Heritage Site.</p>

<figure>
  <img src="pic_trulli.jpg"
alt="Trulli" style="width:100%">
  <figcaption>Fig.1 - Trulli,
Puglia, Italy.</figcaption>
</figure>

</body>
</html>
```

Places to Visit

Puglia's most famous sight is the unique conical houses (Trulli) found in the area around Alberobello, a declared UNESCO World Heritage Site.

Fig.1 - Trulli, Puglia, Italy.

Figure 15. HTML <figure> Element. Source:[author]

The <img> element is used inside the <figure> element to define the actual image or illustration being displayed. This element can be used with or without the <figcaption> element, depending on whether a caption is necessary. By using the <figure>, <figcaption>, and <img> elements together, web developers can create a more organized and visually appealing web page that is easier for readers to navigate and understand.

3.4 HTML Hyperlinks

Hyperlinks, or links, are a fundamental concept of the internet, and understanding what they are requires us to first review the basics of web architecture. Tim Berners-Lee, the creator of the web, outlined the three pillars of web architecture as URL, HTTP, and HTML. These three pillars form the foundation of everything on the internet, from locating and accessing documents to embedded hyperlinks [8].

Prior to the development of the internet, accessing and navigating documents was a difficult task. URLs simplified matters, but manually entering lengthy URLs is cumbersome. This is where hyperlinks come into play. Links allow any text string to be associated with a URL, allowing users to access the target document immediately by activating the link [8].

Links are typically blue and underlined, making them stand out from the surrounding text. To activate a link, users can simply click or tap it. The development of links has revolutionized the internet, making it an incredibly useful tool for accessing and sharing information.

This section will discuss the different types of links and their significance in contemporary web design. Internal links, external links, incoming links, and anchors in HTML will all be explored.

3.4.1 Internal Hyperlink

An internal link connects two pages within the same website. Every website contains internal links unless it is a one-page website. Internal links are essential for improving usability and should be the primary focus when designing a website.

When designing a website, it is important to find the right balance between having too few and too many links. As a rule, whenever a new webpage is added, ensure that at least one of the existing pages links to it. It is counterproductive to link to every page from every other page if a website has more than ten pages. Website navigation design will be discussed in a separate article [8].

3.4.2 External Hyperlink

An external link connects a website to another website. The web is a network of webpages, and external links are essential to provide content that is unavailable on your website.

External links are less important than internal links in the beginning, but they are crucial if a website owner wants search engines to find their website. The more incoming links a website has, the higher its search engine rankings. However, it is unclear how much external links affect the search engine rankings of both the source and target websites [8].

3.4.3 HTML Websites & Search Engines

In the world of websites, links are crucial for users and search engines alike. Search engines use links to index pages by following the links on the page. The visible text of the link helps the search engine determine which search queries are relevant for reaching the target page [8].

Incoming links, also known as backlinks, refer to links from one website to another. This is the opposite of an external link. Website owners are not required to link back when someone links to their website. Anchors, on the other hand, are another type of link that connects two sections of the same document. When a user clicks on an anchor link, their browser navigates to a different section of the same document instead of loading a new one [8].

Search engine rankings are highly valued by companies and websites. The visible text of a link affects which search queries locate a particular URL, while the more incoming links a website has, the higher its search engine rankings. However, it is unclear how much external links affect the search engine rankings of both source and target websites.

It is important to note that the use of links should be strategic and balanced. While internal links improve website usability, too few or too many links can impact the user experience negatively. It is recommended that when adding a new webpage, at least one existing page should link to it. In addition, if a website has more than ten pages, linking to every page from every other page is counterproductive [8].

Overall, links are an essential Internet concept and an integral part of web design and search engine optimization. Proper use of links can improve website usability and search engine rankings, leading to a better user experience and increased traffic.

3.5 URL

URL, short for Uniform Resource Locator, is a fundamental concept in the World Wide Web (WWW). It is a string of text that indicates where a resource, such as a web page, image, or video, can be located on the Internet. URLs are also known as "Web addresses" or "links" in the context of HTTP. When you enter a URL, such as https://developer.mozilla.org, into your browser's address bar, it displays the resource corresponding to that URL [9], URLs are not limited to HTTP and can be employed for file transfer (FTP), email transmission (SMTP), and access to other applications. However, in the context of the WWW, URLs are crucial for retrieving published internet resources, and they are used by browsers to do so.

A valid URL theoretically points to a unique resource, but there are exceptions. For example, a URL pointing to a resource that no longer exists or has relocated. The responsibility of managing both the resource and its associated URL lies with the owner of the web server [9], a URL has several components, some of which are required, while others are optional. The most important sections of a URL are its scheme, authority, resource path, parameters, and anchor. The scheme, the first component of a URL, specifies the protocol that the browser must employ to request the resource. HTTPS or HTTP are the standard protocols for websites [9], The authority component of a URL includes both the domain and port, separated by a colon. The domain identifies the Web server being accessed, and it is usually a domain name, but an IP address can also be used. The port number identifies the "gateway" used to access the web server's resources.

The resource path is the route to the resource on the Web server. It is an abstraction managed by Web servers, devoid of actual substance, and is represented by the "/path/to/myfile.html" component of a URL [9], the parameters component of a URL consists of a collection of key-value pairs separated by the ampersand (&). The web server can use these arguments to perform additional operations before returning the resource [10], the anchor component of a URL is also known as the fragment identifier. It functions as a "bookmark" within a resource, instructing the browser to display the content at the "bookmarked" location [10], there are two types of URLs: absolute and relative. An absolute URL contains all the components required to locate a resource, including the protocol, domain, port, resource path, parameters, and anchor. In contrast, a relative URL uses the context in which it is used to auto complete some components of the URL [9].

URLs are a crucial aspect of the WWW and play a significant role in how users' access and retrieve resources from the internet. They are made up of several components, some of which are required, while others are optional. Knowing the anatomy of a URL can help web developers create efficient and effective websites.

3.6 Web Crawling

Web crawling refers to the process of searching the internet for relevant information, typically using algorithms to locate the most relevant and nearby content. This process is recursive and continues if the results are relevant to the user's interests. Search engines rely on crawlers to retrieve and index newly created and updated web pages using complex

algorithms. Seed URLs, which are a collection of Uniform Resource Locators (URLs), are used to initiate the crawling process [11].

Choosing the starting URL is crucial, as it determines which pages the crawler downloads first. It is generally recommended to use a reputable seed URL, such as those provided by popular search engines like Google or Yahoo. Due to the size of the internet, search engines cannot possibly index every website, and relevant pages should be among the initial few downloads. Each web page's relevance or cost is based on its exceptional quality, its status in terms of outbound connections, and its number of user visits [12].

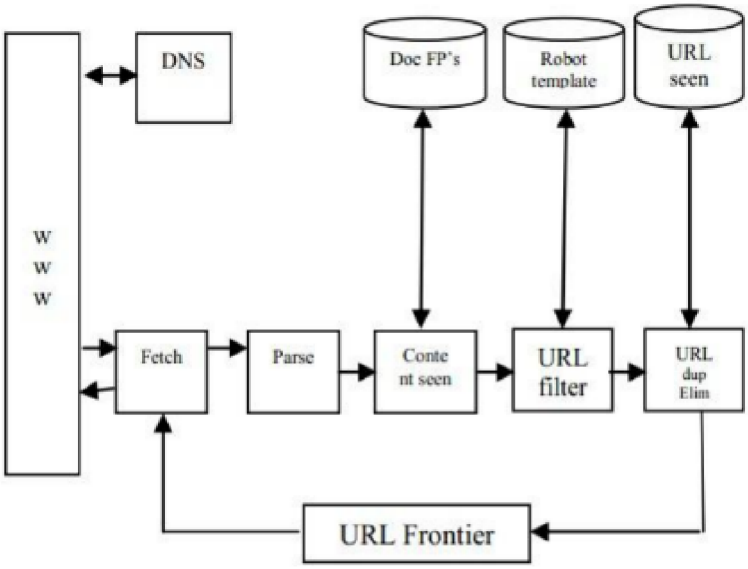

Figure 16. Architecture of a Basic Crawler. Source:[12]

Web crawlers operate in a recursive or loop manner, which involves downloading the website's page, examining the downloaded page, and extracting all of the links, and then repeating the process for each link retrieved. The architecture of a basic web crawler is depicted in Figure 16. Various techniques are employed by search engines to download pages that have been previously downloaded and those that have yet to be downloaded, there are different types of web crawlers, each with its own specific purpose and approach [13].

3.6.1 Focused Crawler

One type is the focused crawler, which is designed to collect web pages that satisfy a specific property by carefully prioritizing the crawl frontier and managing the hyperlink exploration process. For example, a focused crawler can be set up to "crawl pages with high PageRank" or "crawl pages about baseball." A focused crawler determines the likelihood of a page being relevant before downloading an unvisited page, making it more efficient than conventional methods [14].

3.6.2 Incremental Crawler

Another type of web crawler is the incremental crawler, which collects modifications made since the previous crawl. An incremental crawler returns to the web at regular intervals to update its collection. However, the freshness of content decreases with each

incremental crawl run, as the crawler needs to catch up with the content being updated faster than it can crawl them. If content output continues, incremental crawling ultimately results in updating content created during incremental crawling. When the rate of content production slows, the incremental crawl will catch up, and the interval becomes irrelevant because the incremental crawl will take as much time as necessary to complete.

3.6.3 Continuous Crawler

A continuous crawler is designed to keep a search index more current. The frequency of requests increases, but the maximum number of concurrent requests on a single repository/host continues to be governed by "Crawl Impact Rules," which specify the maximum number of concurrent threads that can make requests. The continuous crawl footprint resembles the incremental crawl footprint, and multiple continuous crawls can run concurrently for the same content source and continuously update the index.

3.6.4 Parallel Crawler

A parallel crawler is a crawler that simultaneously executes multiple processes, aiming to maximize download speed while minimizing parallelization overhead and avoiding frequent downloads. Multiple crawling procedures constitute a parallel crawler. Each process performs the fundamental duties of a single process crawler, including downloading web pages, storing them locally, extracting URLs from downloaded pages, and then following links.

3.6.5 Distributed Crawler

Distributed web crawling is a distributed computing technique in which multiple computers are used by Internet search engines to index the Internet via web crawling. Such systems may enable users to volunteer their computing and bandwidth resources for web page crawling. By distributing these tasks across multiple computers, costs associated with maintaining large computing clusters are avoided [15].

3.7 Web Crawling Algorithm

Web crawling algorithms play a critical role in the effectiveness of web crawlers. The primary objective of these algorithms is to identify and download the most important web pages first. A good web crawling algorithm should not only retrieve relevant pages but also high-quality pages. Therefore, developers must consider the algorithm used when designing web crawlers.

3.7.1 Breadth First Search (BFS)

The Breadth First Search (BFS) algorithm is one of the most used algorithms in web crawling. The main objective of this algorithm is to download the most important pages first. It does so by exploring all adjacent URLs at the same level, starting from the root URL, and then descending to the next level until the desired result is found. The BFS algorithm is particularly useful when the target is located on the lower branches of a taller tree [28].

However, BFS may not always be the most efficient strategy. For example, in game trees with many branches, such as a chess game, where every route leads to the same goal and is the same length, BFS may not work well. In such cases, other algorithms like Depth First Search (DFS) or the A* algorithm might be more effective.

In the BFS algorithm, the search starts with the root URL and collects all adjacent URLs at the same level. The search then proceeds to download the equivalent versions of each URL. For example, if the root URL is "www.example.com," the BFS algorithm will download URLs I, II, and III, and then proceed to download I.a, I.b, I.c, II.a, II.b, III.a, and III.c. Additionally, it will download i.a.a, i.a.b, i.a.c, ii.a.a, ii.a.b, and iii.a.c as shown in Figure 17.

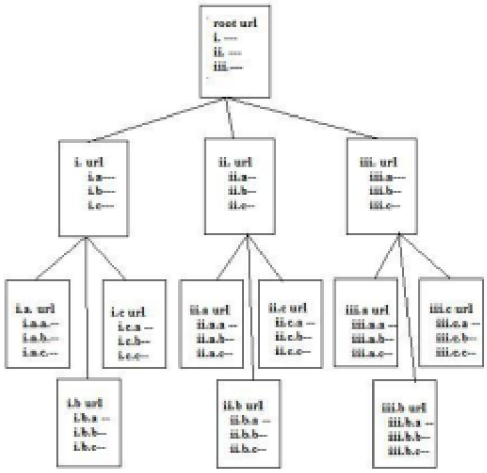

Figure 17. Architecture of a Breath-First Search Algorithm. Source:[25]

3.7.2 Depth First Search

The depth-first search algorithm is an exhaustive search technique that starts with the parent URL and probes deeper into the child URLs. The algorithm prioritizes the leftmost child first, and if there are multiple children, it proceeds to the next child until there are none left. It then backtracks to the next unexplored node, and the procedure is repeated. This method ensures that each edge is visited once in breadth. However, if the branches are too large, the algorithm may get stuck in an infinite loop [16].

The depth-first search algorithm is a useful strategy for certain search problems, but it may not be the most effective approach for web crawling. In our case, after querying the root URL, the initial URL I is downloaded. Then, I.A is downloaded first, followed by I.A.A, I.A.B, and I.A.C. Next, I.B and its child and I.C and its child are downloaded in subsequent steps.

The depth-first search algorithm can be advantageous when the target is located deep within the tree structure. It can also be helpful in searching game trees with few branches. However, it may not be the best choice for web crawling because it is more likely to get stuck in an infinite loop if there are too many branches. In contrast, the breadth-first search algorithm is more efficient in searching for web pages that are closer to the root URL.

3.7.3 Page Rank

The Page Rank algorithm is an important search algorithm that considers the degree of inter-connectedness between web pages. This algorithm assumes that the more incoming links a web page has, the more valuable and important it is. The basic concept behind Page Rank is that a page that has more inbound links from other high-quality web pages should be ranked higher in search engine results pages (SERPs) than a page with fewer inbound links [17].

Page Rank is calculated using a formula that considers the Page Rank of each webpage that links to the page being ranked. In the formula, the Page Rank of page P1 is equal to the sum of the Page Rank of each webpage that links to page P1 divided by the number of outbound links on that page. This calculation is repeated for all web pages in the network, and the final Page Rank is determined by iteratively computing the values until convergence is achieved [17].

One of the advantages of Page Rank is that it considers the quality of the web page that links to the page being ranked. The Page Rank of a web page is determined by the sum of the Page Ranks of the web pages that link to it. Therefore, a page that is linked to by high-quality web pages will have a higher Page Rank than a page that is linked to by low-quality web pages.

Another advantage of Page Rank is that it can be used to discover the most important web pages in a network. By analysing the link structure of a network, Page Rank can identify the web pages that are most likely to be relevant to a given search query. This makes it an important tool for search engine optimization (SEO) and digital marketing.

3.7.4 Path-Ascending Algorithm

The path-ascending algorithm is a web crawling algorithm that involves crawling each route from the main page to the final file of that URL. This algorithm enables a crawler to obtain additional data from a site more easily. When given the seed URL, the crawler ascends to each path in each URL (Uniform Resource Locator) that it intends to crawl. For example, when given the seed URL https://abdulbasitaliyu.com/, the crawler would attempt to crawl /abdulbasitaliyu.com/, /user/, and /profile.html.

The advantage of using a path-ascending crawler is that it can find isolated resources or resources without an inbound link that a standard crawler would have missed. This is particularly useful when trying to obtain information that is not readily accessible through a regular web search. For instance, a path-ascending crawler could be useful in gathering data from a website that has hidden files or directories.

The path-ascending algorithm is also useful when trying to obtain a complete picture of a website's structure. By crawling each route from the main page to the final file of that URL, the crawler can generate a complete site map. This site map can be useful for website owners and administrators as it enables them to identify potential issues with their site's structure, such as broken links or missing pages [18].

3.7.5 Genetic Algorithm

The genetic algorithm is a search algorithm that emulates the process of biological evolution. It selects the fittest offspring by crossing the children of the best individuals in a population. The process is guided by a fitness function that evaluates the potential solutions to a problem. Genetic algorithms are particularly useful in situations where the user has limited time to search through a massive database. They are also effective in producing multimedia results [19].

Unlike traditional search algorithms that start their search from a single location, genetic algorithms search throughout a whole population. This feature contributes to their algorithmic resilience, making them capable of handling complex search problems. Genetic algorithms have a wide range of applications, including optimization, machine learning, image processing, and data analysis.

In a genetic algorithm, the initial population is randomly generated, and everyone in the population represents a potential solution to the problem. The fitness function evaluates everyone based on how well it satisfies the constraints of the problem. The individuals with the highest fitness scores are selected for reproduction, and their genetic material is combined to produce offspring. This process is repeated until the desired solution is found.

One advantage of genetic algorithms is that they are capable of handling multiple solutions to a problem. They can identify the optimal solution in a predetermined amount of time, even when dealing with large and complex datasets. However, genetic algorithms can be computationally intensive, especially when dealing with large populations. Therefore, their performance depends on the available computational resources.

3.8 A Comparative Analysis of Web Crawlers and Crawling Algorithms

The act of crawling can potentially improve the overall quality of a search engine's service. It is imperative to utilize optimal crawlers and crawling algorithms to accurately assess the quality and up to date-ness of web pages. To this end, I have conducted extensive research on the different types of web crawlers and crawling algorithms, analysing their respective advantages, drawbacks, as well as strengths and weaknesses [19].

Table 1 Advantages and Disadvantages of Crawler

Crawler	Advantages	Disadvantages
Focused Crawler	Requires less time and effort for web page processing	The zero-probability problem and determining the relevance of unvisited URLs
Incremental Crawler	Allow revisitation of pages at different rates	Allow Significant alteration will not degrade freshness
Continuous Crawler	Allow Changes will continue to be handled concurrently.	Slightly increase the host's burden
Parallel Crawler	Network load balancing	Require backup storage

Distributed Crawler	Reduces hardware requirements and improves download speeds and reliability	Web partitioning/partitioning and placement of data centers are necessary

Table 2 Pros and Cons of Crawling Algorithms

Crawler Algorithm	Pros	Cons
Breadth First Search	Designed for circumstances in which objectives is in shallower portions of a taller tree	Not as effective when there are numerous branches on the objective tree
Depth First Search	Only requires storage of visited pages in a single web graph	When the branches are extensive, an infinite loop may result
Continuous Crawler	Allow Changes will continue to be handled concurrently.	Slightly increase the host's burden
Page Rank	In the extremely limited time, essential pages are downloaded-In a very short period, vital pages are always downloaded in high	Create a lot of supplementary pages and links.
Path Ascending	It can discover resources that a typical crawler would have overlooked, such as those that are isolated or have no inbound links	crawl each path from the home page to the very last file associated with that URL
Genetic	An optimization approach to the search	Weighted choice of characteristics

3.9 Web Crawling Algorithm

Various algorithms and metrics have been proposed to direct a web crawl towards pages of high quality. In this section, we will discuss some related work in this field. Shaojie Qiao [20] proposed SimRank, a new algorithm for ranking web pages based on a similarity measure from the vector space model. They proposed a new measure of similarity to compute the similarity of pages and apply it to divide a web database into various web social networks (WSNs). The effectiveness of SimRank has been proven in various applications, such as clustering, keyword search, and entity search.

Tian Chong [34] proposed a new type of algorithm for page ranking by combining the classified tree with the static algorithm of PageRank. This approach allows the classified tree to be constructed based on many users' similar searching results and can reduce the problem of Theme-Drift, which is caused by using PageRank alone, and the problem of outdated web pages. This approach has the potential to increase the efficiency and effectiveness of search engines.

In [22], Pann Yu Mon, Chew Yew Choong, and Yoshiki Mikami proposed a crawler focused on the Myanmar language along with its architecture and performance. LSC's primary function is to identify the various encodings of Myanmar web pages. The LSC has successfully downloaded a sufficient number of Myanmar web pages through experimentation. Several metrics, including the accuracy rate of the language identifier, its performance, and its recall rate, are presented.

Saeko Nomura, Satoshi Oyama, and Tetsuya Hayamizu proposed integrating two improvement methods in [23]. This method yields positive results. Not only can relevant pages be extracted, but those pages can also be loaded on principal eigenvectors or non-principal vectors at a low computational cost for topics with a broader scope. This approach can improve the efficiency of web crawling and ensure that the most relevant and high-quality pages are collected.

Wenxian Wang, Xingshu Chen, Yongbin Zou, Haixhou Wang, and Zongkun Dai [37] proposed an effective crawler based on Naive Bayes to collect a large number of relevant pages for hierarchical website layouts. The proposed approach considers the hierarchical structure of websites and uses a two-level classification method to classify web pages. The results demonstrate that the proposed crawler can achieve high precision and recall rates while minimizing the number of irrelevant pages.

3.10 Summary

Web crawling algorithms are essential tools for extracting valuable data from the internet. They work by following hyperlinks on web pages and collecting data from the pages they visit. The key components and functionalities of a web crawling algorithm depend on the specific goals of the task at hand. In the case of retrieving articles related to a particular topic, the algorithm should be designed to identify and extract relevant information from web pages.

Existing solutions for web crawling algorithms have been developed for various purposes, including search engine indexing, web content mining, and data extraction. These solutions include rule-based systems, heuristic algorithms, and machine learning techniques. Rule-based systems rely on a set of predefined rules for identifying relevant data. Heuristic algorithms use statistical analysis and machine learning techniques to identify patterns in data. Machine learning techniques use algorithms that can learn from data and improve over time.

The effectiveness of a web crawling algorithm depends on its ability to retrieve relevant data. The proposed algorithm for retrieving articles related to the agrarian sector has several components, including the identification of relevant websites, the extraction of relevant information from web pages, and the use of natural language processing techniques to filter out irrelevant content. The algorithm also includes the ability to follow links to related pages and crawl through multiple levels of the website.

Implementing the proposed algorithm requires the use of appropriate technology. Python, Scrapy, and Beautiful Soup are popular technologies for web crawling and data extraction. Scrapy is a powerful and flexible web crawling framework that provides features such as automatic request throttling, cookie handling, and support for multiple types of data storage.

3.11 Challenges

One challenge of implementing a web crawling algorithm is dealing with dynamic content, such as JavaScript-based page elements, which can cause problems for traditional web crawlers. Other challenges include dealing with duplicate content, handling errors and exceptions, and managing large amounts of data.

Another challenge is how to choose the method for removing or retiring unwanted pages. Web crawlers often collect pages that are irrelevant or outdated, and removing or retiring these pages is essential to ensure the freshness and accuracy of the collected data. However, there is no one-size-fits-all approach for removing unwanted pages, and different methods may be needed for different types of data.

The practical part of this project differs from related work in that it focuses specifically on retrieving articles related to the agrarian sector. The proposed algorithm is designed to extract information such as category, headline, URL, thumbnail, and summary from web pages. The algorithm also includes the ability to follow links to related pages and crawl through multiple levels of the website dynamically.

In summary, web crawling algorithms are powerful tools for extracting valuable data from the internet. The key components and functionalities of a web crawling algorithm depend on the specific goals of the task at hand. The proposed algorithm for retrieving articles related to the agrarian sector includes the ability to identify relevant websites, extract relevant information from web pages, and follow links to related pages dynamically. Implementing the proposed algorithm requires the use of appropriate technology, and challenges include dealing with dynamic content, duplicate content, errors and exceptions, and managing large amounts of data. The practical part of this project differs from related work in its focus on retrieving articles related to the agrarian sector and its use of dynamic content crawling techniques.

4 Practical Part

4.1 Research Design

In the practical component of the thesis, a web crawling algorithm was designed and implemented using the Python programming language and the Scrapy framework. The algorithm was designed to crawl web articles pertaining to the agriculture industry with the purpose of collecting current and relevant information on the agricultural industry for study and analysis.

The Scrapy framework, a popular web scraping and crawling tool, was utilised for this objective, providing consumers with numerous advantages. It offers a full range of capabilities for managing many elements of web crawling, such as managing HTTP requests and answers, parsing HTML and XML documents, and storing crawled data. In addition, it offers concurrency and asynchronous processing, which can improve the crawling process's effectiveness and speed.

To validate the algorithm in the context of English-language websites, it was tested on a number of prominent agricultural-related websites. The start urls for the web crawling included 'https://www.fwi.co.uk/news', 'https://www.farmprogress.com/', 'https://www.agweek.com/', 'https://www.agriculture.com/', 'https://www.agweb.com/', and 'https://www.agdaily.com/'. The algorithm was created to prioritise specific websites and domains that are known to have a considerable volume of content important to the agricultural industry. In addition, it was optimised to prioritise the crawling of high-quality pages, such as those with high page ranks or agrarian-specific keywords.

4.2 Data Collection

The collecting of data is a crucial element of any research, and it is necessary to guarantee that the data collected are correct and trustworthy. For this study, data gathering involves the automated web crawling of agrarian sector-related web articles using a custom-designed and implemented algorithm. The major objective was to assess the algorithm's performance in obtaining articles pertinent to the agricultural sector.

To assure the quality of the obtained data, the algorithm was developed to automatically search and collect data from English-language agrarian-related websites. The collected data was recorded in a JSON file with a structured manner for easy access and analysis. In addition, a pilot study was done to test and confirm the algorithm's performance in retrieving agrarian-sector-relevant publications.

Due to IP blocking and ethical reasons, the pilot study will be limited to retrieving only 5 articles from each of the above English-language websites pertaining to the agrarian sector. The collected data will be analysed using the same quantitative and qualitative methods described above in order to evaluate the algorithm's effectiveness in retrieving relevant and current information pertaining to the agrarian sector. During the data gathering process, ethical issues will continue to be taken into account to guarantee that the collected data is free of biases and inaccuracies. The purpose of this study is to validate the algorithm's efficacy in the context of English-language agricultural websites.

Standard measurements of an algorithm's performance in retrieving relevant articles, precision and recall were calculated as part of quantitative analysis. By comparing the retrieved articles to a manually curated list of relevant articles, precision and recall were computed. In qualitative analysis, the content of the collected articles was analysed to establish their relevance to the agricultural industry….

4.3 Algorithm Design

The Scrapy framework and Python were utilised to collect, pre-process, filter, and extract pertinent information from websites associated with the agricultural industry. The algorithm was design with a number of factors in mind to prioritise the retrieval of relevant articles and prevent retrieving irrelevant pages.

Some design choices were made in order to maximise the efficiency of the algorithm. For example, the algorithm was built to avoid crawling unnecessary pages and prioritise the collecting of data from trustworthy sources. The algorithm also takes privacy and data protection rules governing website owners into account.

The algorithm was created with a straightforward and simplified approach, employing basic technology to make its replication straightforward. The resulting algorithm was rigorously tested and validated to assure its efficacy in gathering data pertinent to the agricultural sector. To adjust the algorithm's behaviour, no extra features or tools were added.

4.4 Implementation

In this thesis the implementation of the web crawling algorithm which aimed to select a technology that would enable efficient and precise data collecting and to evaluate the algorithm in the context of English-language websites. Due to its widespread use and popularity as an open-source web crawling platform, the Scrapy framework was selected as the technology for algorithm development. Scrapy streamlines the process of gathering data from web pages and offers a high-level interface that facilitates the building of robust and scalable web scraping applications, making it an ideal contender for the development of the web crawling algorithm in this thesis.

Python, the programming language used to create the web crawling technique, was installed on the operating system in order to establish the development environment. At the time of writing, the most recent version of Python available was version 3, which was installed using the command "sudo apt-get install python3" on the terminal.

Scrapy was installed on the operating system with the command "sudo apt-get install scrapy" in the terminal, allowing the author to collect data from websites using the Scrapy framework. In addition to installing Scrapy on the operating system, any essential packages for the implementation process were also installed.

As a text editor, the author utilised Microsoft Studio Code (VS Code). Due to its user-friendly design and support for multiple programming languages, VS Code is a popular Integrated Development Environment (IDE) (IDE). But, developers can choose from a variety of additional integrated development environments based on their

preferences, the algorithm for web crawling was carefully and effectively built, taking into account the partial objectives of selecting an appropriate technology and verifying the algorithm in the context of English-language websites. In order to establish the algorithm's efficacy in obtaining data important to the agriculture industry, it was tested and validated.

4.5 AgrarianSpider Web Crawling

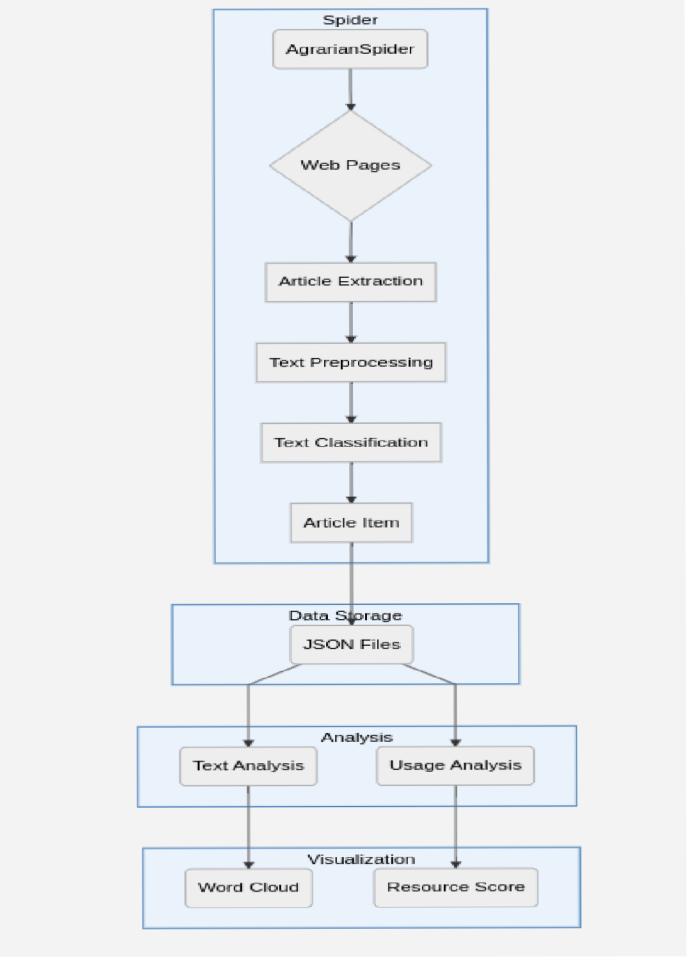

Figure 18. Architecture Component Diagram of theAgrarianSpider Algorithm. Source:[author]

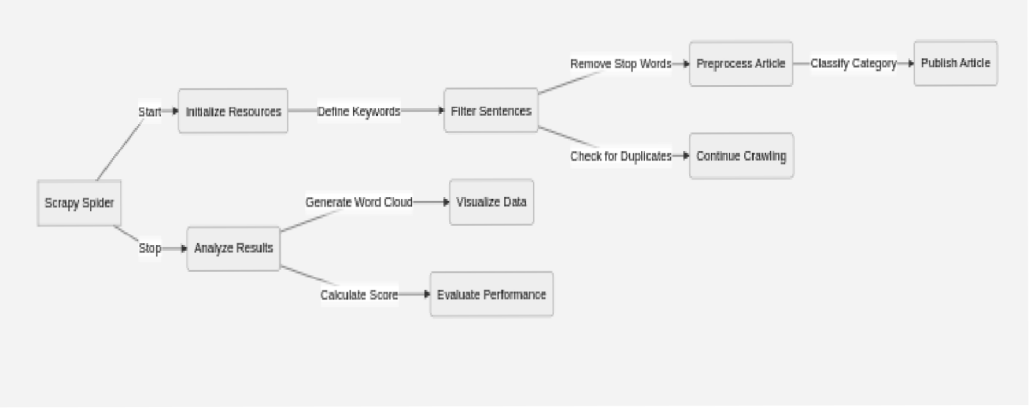

Figure 19. Architecture Component Diagram of the AgrarianSpider Algorithm 2. Source:[author]

In *Figure 18 & 19* depicts the various components of the Agrarian Spider and their interrelationships. The Scrapy Spider is the primary component that controls web page crawling and parsing. In the "Initialize Resources" block, the relevant resources are initialised, the "Filter Sentences" block accepts sentences from the web sites and filters them based on the defined agrarian-related keywords, phrases, and synonyms. The filtered sentences are then preprocessed in the "Preprocess Article" block, where stop words and special characters are filtered out.

Based on the parsed content, the "Classify Category" section labels the article as "Agriculture" or "Livestock." If the article is determined to be pertinent, it is published under the "Publish Article" section. The spider continues crawling in the "Continue Crawling" block if it is a duplicate, After all web pages have been crawled and processed, the results are analysed in the "Analyze Results" section. In the "Visualize Data" block, the spider constructs a word cloud of the processed text, and in the "Evaluate Performance" block, it evaluates the performance. The latter determines the score for each record based on elapsed time, CPU utilisation, memory utilisation, and network utilisation.

This component diagram provides a comprehensive understanding of the Agrarian Spider's various stages and how they assist to reaching its goals.

```
import datetime
import scrapy
import time
import psutil
import os
import json
from scrapy.exceptions import DropItem
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import re
from wordcloud import WordCloud
from matplotlib import pyplot as plt
class AgrarianSpider(scrapy.Spider):
```

Source code 1 AgrarianSpider – partital component 1

```
name = 'agrarian_spider'
    allowed_domains = ['www.fwi.co.uk', 'www.farmprogress.com', 'www.agweek.com',
                       'www.agriculture.com', 'www.agweb.com', 'www.agdaily.com']
    start_urls = ['https://www.fwi.co.uk/news', 'https://www.farmprogress.com/',
'https://www.agweek.com/',
                  'https://www.agriculture.com/', 'https://www.agweb.com/', 'https://www.agdaily.com/']
    max_articles = 5
    custom_settings = {
        'FEEDS': {
            'articles00109.json': {
                'format': 'json',
                'encoding': 'utf9',
                'store_empty': False,
                'fields': None,
                'indent': 4,
                'item_export_kwargs': {
                    'export_empty_fields': True,
                },
            },
        },
```

Source code 2 AgrarianSpider – partital component 1

```
        'CONCURRENT_REQUESTS': 8,
        'DOWNLOAD_DELAY': 3,
        'USER_AGENT': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/59.0.3029.110 Safari/537.3',
        'ROBOTSTXT_OBEY': True,
        'ROTATING_PROXY_LIST_PATH': 'proxy.txt',
        'DOWNLOADER_MIDDLEWARES': {
            'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware': 110,
            'rotating_proxies.middlewares.RotatingProxyMiddleware': 610,
            'rotating_proxies.middlewares.BanDetectionMiddleware': 620,
        }
    }

    def __init__(self, *args, **kwargs):
        super(AgrarianSpider, self).__init__(*args, **kwargs)
        self.stop_words = set(stopwords.words('english'))
```

Source code 3 AgrarianSpider – partital component 2

This portion of code above *Source code 1, 2 & 3* defines the AgrarianSpider class, which is a descendant of the ScrapySpider class. The objective of this spider is to crawl through six agrarian-related websites, including www.fwi.co.uk, www.farmprogress.com, and www.agdaily.com, in search of articles containing relevant keywords, phrases, and synonyms.

Several Python libraries are imported, including datetime, scrapy, time, psutil, os, json, re, and matplotlib.pyplot. It also imports the stopwords and word tokenize functions from the Natural Language Toolkit (NLTK) library as well as the WordCloud class from the wordcloud library. Many properties are defined for the AgrarianSpider class, including the spider's name, permissible domains, start URLs, maximum amount of articles to extract, and custom settings. Custom configurations include characteristics such as feed format, concurrent requests, download latency, user agent, and middleware for managing rotating proxies.

The init method of the AgrarianSpider class initialises an instance of the class, sets the stop words attribute to a list of English stopwords using the stopwords.words('english') function from NLTK, and calls the init method of the Scrapy Spider class to perform any additional spider initialization.

```
def parse(self, response):
        start_time = time.time()
        start_cpu = psutil.cpu_percent()
        start_mem = psutil.virtual_memory().used
        start_sent = psutil.net_io_counters().bytes_sent
        start_recv = psutil.net_io_counters().bytes_recv


        # Define more specific keywords related to the agrarian sector
        keywords = ['agriculture', 'farming', 'crop production', 'livestock farming',
                    'harvesting', 'rural economy', 'soil management', 'agronomy',
                    'precision farming', 'sustainable agriculture']


        # Use phrases that are commonly used in the agrarian industry
        phrases = ['crop yields', 'livestock management', 'irrigation systems',
                   'pest control', 'fertilizer application', 'tractor technology',
                   'agricultural machinery', 'seed varieties', 'agribusiness']
```

Source code 4 AgrarianSpider – parse method 1

```
    # Use synonyms of the keywords
    synonyms = ['agronomic', 'farms', 'cultivation', 'cattle', 'dairy farming',
                'animal husbandry', 'ranching', 'small-scale farming', 'organic farming',
                'greenhouse production', 'food security']
    count = 0
    # Find sentences containing the keywords, phrases, or synonyms
    for sentence in response.xpath('//text()').extract():
        for term in keywords + phrases + synonyms:
            if term in sentence.lower():
                if not self.is_duplicate(sentence):
                    # Add the processed text and category to the item
                    item = {'website': response.url,
                            'article': sentence.strip()}
                    processed_item = self.process_item(item)
                    if processed_item is not None:
                        yield processed_item
                        count += 1
                        if count >= self.max_articles:
                            return
    # Follow links to other pages of the website
    for href in response.xpath('//a/@href'):
        url = response.urljoin(href.extract())
        if any(domain in url for domain in self.allowed_domains):
            yield scrapy.Request(url, callback=self.parse)
# Calculate resource usage
    end_time = time.time()
    end_cpu = psutil.cpu_percent()
    end_mem = psutil.virtual_memory().used
    end_sent = psutil.net_io_counters().bytes_sent
    end_recv = psutil.net_io_counters().bytes_recv
    elapsed_time = end_time - start_time
    cpu_usage = end_cpu - start_cpu
    mem_usage = end_mem - start_mem
    net_sent = end_sent - start_sent
    net_recv = end_recv - start_recv


    # Save resource usage to a JSON file
    usage = {
        'elapsed_time': elapsed_time,
        'cpu_usage': cpu_usage,
        'mem_usage': mem_usage,
        'net_sent': net_sent,
        'net_recv': net_recv,
    }
    with open('usage109.json', 'a') as f:
        f.write(json.dumps(usage) + '\n')
```

Source code 5 AgrarianSpider – parse method 2

The section of code *source code 4 & 5* defines a method named "parse" within the "AgrarianSpider" class. This method is used in web crawling to collect information from online pages by following hyperlinks and pulling data from pages that match criteria.

In the beginning of the process, numerous performance measures, including CPU and memory consumption, as well as network data, are collected to measure the spider's resource utilisation. Thereafter, keywords, phrases, and synonyms linked to the agrarian sector are defined and used to locate relevant publications.

The method then iterates over each sentence on the page and determines whether it contains any of the defined keywords, phrases, or synonyms. If a sentence contains one of these terms, it is reviewed for duplicate material, and if not, it is processed and classified into one of two categories: Agricultural or Livestock. The processed item is subsequently added to the output item collection.

The procedure then looks for hyperlinks on the page and follows them to other online pages. This is done to ensure that all web pages satisfying the criteria are crawled and included in the results, the procedure concludes by calculating the spider's resource consumption and storing the results in a JSON.

```
def process_item(self, item):
    # Remove special characters and numbers from the article text
    text = re.sub('[^A-Za-z]+', ' ', item['article'])

    # Tokenize the article text
    tokens = word_tokenize(text)

    # Remove stop words from the tokenized text
    filtered_tokens = [
        token for token in tokens if not token.lower() in self.stop_words]

    # Join the filtered tokens back into a string
    processed_text = ' '.join(filtered_tokens)

    # Classify the article based on the processed text
    if 'agriculture' in processed_text or 'farming' in processed_text:
        item['category'] = 'Agriculture'
    elif 'livestock' in processed_text or 'cattle' in processed_text:
        item['category'] = 'Livestock'
    else:
        return None

    # Add the processed text to the item
    item['processed_text'] = processed_text
```

Source code 6 AgrarianSpider – process_item method

The preceding code *source code 6* fragment defines the process item() method, which is invoked by the parse() method of the AgrarianSpider class. This function is responsible for processing the collected article text from the internet and classifying it into either the Agricultural or Livestock category.

Using regular expression, the procedure first removes all special characters and digits from the article text. The text is then tokenized with the word tokenize () function of the nltk toolkit. Using a list of predefined stop words, which is defined in the __init__() function of the AgrarianSpider class, the next step is to remove any stop words from the tokenized text.

After removing stop words, the join() technique is used to rejoin the filtered tokens into a string. Based on the existence of particular keywords, the resultant string is then used to classify the article into one of two categories. If either 'agricultural' or 'farming' appears in the processed text, the article is classed as 'Agriculture'. If the item contains the terms "livestock" or "cattle," it is classed as "Livestock." If none of these keywords are present in the text being analysed, the procedure returns Nothing, the processed text and category information are then added to the item dictionary, which is returned to the parse () method.

```
def is_duplicate(self, sentence):
    # Check if the sentence has already been extracted
    filename = 'articles00109.json'
    if os.path.exists(filename):
        with open(filename, 'r') as f:
            data = f.read()
            if sentence in data:
                return True
    return False
```

Source code 7 AgrarianSpider – is_duplicate method

The is_duplicate method *source code 7* of the AgrarianSpider class determines whether a particular sentence has been taken from a webpage and saved in the articles00109.json file previously. This is done to prevent the extraction and storage of redundant data.

Using the os.path.exists() method, the method verifies whether the file articles00109.json already exists in the current directory. If the file exists, the file is opened in read mode using the with open() command and its contents are read into the data variable. The method then determines whether the specified sentence already exists in the data variable, indicating that it has been extracted and stored. If the sentence is found, the procedure returns True to indicate that the sentence already exists. If the sentence is not discovered, the procedure returns False, indicating that the sentence has never been extracted and is unique.

```
def score_article(self, article):
    if not isinstance(article, dict):
        return 0
    score = 0
    # Check if the article contains any of the top keywords
    top_keywords = ['agriculture', 'farming', 'livestock',
                    'cattle', 'harvesting', 'soil management']
    for keyword in top_keywords:
        if keyword in article['processed_text']:
            score += 5
    # Check if the article contains any of the secondary keywords
    secondary_keywords = ['crop production', 'rural economy',
                          'agronomy', 'precision farming', 'sustainable agriculture']
    for keyword in secondary_keywords:
        if keyword in article['processed_text']:
            score += 3
    # Check if the article contains any of the relevant phrases
    relevant_phrases = ['crop yields', 'livestock management',
                        'irrigation systems', 'pest control', 'fertilizer application']
    for phrase in relevant_phrases:
        if phrase in article['processed_text']:
            Score += 2
# Check if the article comes from a credible source
    credible_sources = ['www.fwi.co.uk/news', 'www.farmprogress.com', 'www.agweek.com',
                        'www.agriculture.com', 'www.agweb.com', 'www.agdaily.com']
    if article['website'] in credible_sources:
        score += 5
    # Check if the article was published recently
    if article['date'] >= datetime.now() - datetime.timedelta(days=7):
        score += 3
    return score
```

Source code 8 AgrarianSpider – score_article method -parital component

Next part *source code 8* introduces a function named score article that accepts an article as input and returns a score based on a set of criteria. If the input is not a dictionary, the method returns 0; otherwise, it returns 1. Afterwards, the function computes the article's score depending on a variety of parameters.

Initially, the function determines whether the article contains any of the top keywords associated with the agricultural industry, such as 'agriculture,' 'farming,' 'livestock,' 'cattle,' 'harvesting,' and' soil management.' The function adds 5 points to the score if any of these terms appear in the article, the function then determines whether the article contains any of the secondary keywords, such as 'crop production,' 'rural economy,' 'agronomy,' 'precise farming,' and' sustainable agriculture' The function adds three points to the score if the article contains any of the specified keywords.

The function then determines whether the article contains relevant phrases, such as "crop yields," "livestock management," "irrigation systems," "pest control," and "fertiliser application." The function adds two points to the score if any of these phrases appear in the article, the function additionally verifies that the article originates from a reputable source, as stated in the credible sources list, and adds 5 points to the score if the article's website is on the list, the function concludes by adding three points to the score if the article was published within the past week. The score article function calculates an overall score for an article based on its relevance to the agriculture sector, its content, its source, and its publication date.

```
def closed(self, reason):
    # Analyze and visualize the extracted data
    filename = 'articles00109.json'
    if os.path.exists(filename):
        with open(filename, 'r') as f:
            data = f.read()
            articles = json.loads(data)
            # Extract the processed text from the articles
            processed_text = [article['processed_text']
                              for article in articles]


            # Score each article
            scores = []
            for text in processed_text:
                score = self.score_article(text)
                scores.append(score)
            # Write scores to a JSON file
            with open('scores109.json', 'w') as f:
                json.dump(scores, f)


            # Join the processed text into a single string
            text = ' '.join(processed_text)
            # Generate a word cloud from the text
            wordcloud = WordCloud(width=900, height=400,
                                  background_color='white').generate(text)
            # Plot the word cloud
            plt.figure(figsize=(10, 5))
            plt.imshow(wordcloud, interpolation='bilinear')
            plt.axis('off')
            plt.show()
    else:
        self.logger.error(f'File {filename} not found')
            'net_usage': 0.2,
        }
```

Source code 9 AgrarianSpider – close method -parital component 1

```
    filename1 = 'usage109.json'
    if os.path.exists(filename1):
        # Load the JSON usage result from file
        with open(filename1, 'r') as f:
            data = json.load(f)


        # Calculate the max elapsed time, max memory usage, and max network usage
        max_elapsed_time = max(record['elapsed_time'] for record in data)
        max_mem_usage = max(record['mem_usage'] for record in data)
        max_net_usage = max(record['net_sent'] +
                            record['net_recv'] for record in data)


        # Define the weights for each criterion
        weights = {
            'elapsed_time': 0.3,
            'cpu_usage': 0.2,
            'mem_usage': 0.3,

# Calculate the score for each record
        for record in data:
            score = 0
            score += weights['elapsed_time'] * \
                (max_elapsed_time - record['elapsed_time']) / max_elapsed_time
            score += weights['cpu_usage'] * (1 - record['cpu_usage'] / 100)
            score += weights['mem_usage'] * \
                (1 - record['mem_usage'] / max_mem_usage)
            score += weights['net_usage'] * \
                (record['net_sent'] + record['net_recv']) / max_net_usage
            record['score'] = score


        # Export the results to a new JSON file
        with open('usage_scored109.json', 'w') as f:
            json.dump(data, f, indent=4)
    else:
        self.logger.error(f'File {filename1} not found')
```

Source code 9 AgrarianSpider – close method -parital component 2

Finally, the closed method source code 7 & 8 is a built-in Scrapy method that is executed when all pages have been scrapped. The approach analyses the extracted data, builds a word cloud to show the captured articles' keywords, and evaluates the spider's usage during the scraping process. The technique begins by determining whether or not the file with the extracted articles exists. If the file exists, the content of the file is loaded, the processed text is extracted from the articles, and each article is scored based on specific criteria, such as the presence of primary and secondary keywords, relevant phrases, the credibility of the source, and the recency of the publication. The scores are then written to a new JSON file.

Additionally, the approach combines the processed text into a single string, builds a word cloud from the text, and displays the word cloud using the WordCloud and matplotlib packages. The word cloud illustrates the most prevalent search terms taken from the articles, with more frequent terms appearing larger. Lastly, the method verifies the existence of the file that stores spider usage data. If the file exists, the content is loaded and the maximum elapsed time, memory consumption, and network utilisation are calculated. The system then assigns weights to each criterion and assigns a score to each record. The scores are then exported to a new JSON file, with each item including usage information and the associated score.

5 Results and Discussion

5.1 Results

Observable based on the results generated by the Spider are the following:

The 'articles00109.json' figure 20 file contains information taken from five websites pertaining to agriculture and animals by the Spider.

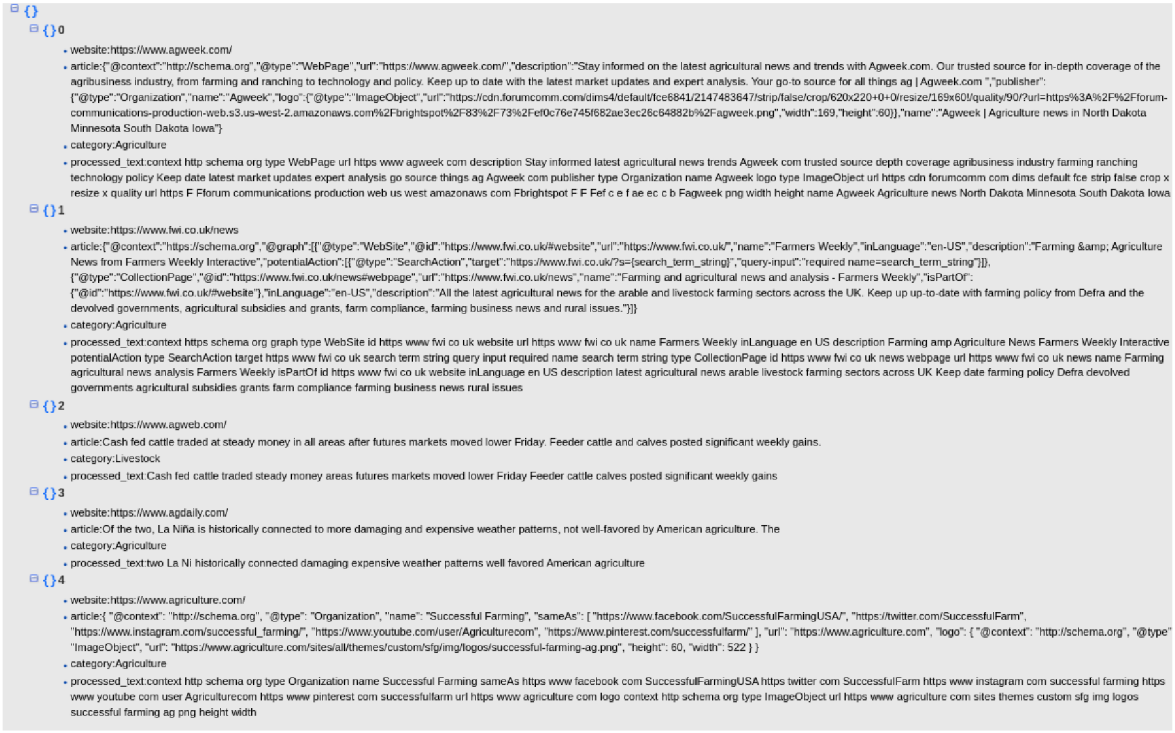

Figure 20. Generated articles00109.json result from AgrarianSpider Algorithm. Source:[author]

The Spider also assessed the relevance of the extracted articles' content to agriculture, livestock, and related keywords. A scoring system that allocated points for the existence of keywords, phrases, and sources generated the scores. The scores are saved in the file'scores109.json' as show in the figure 20

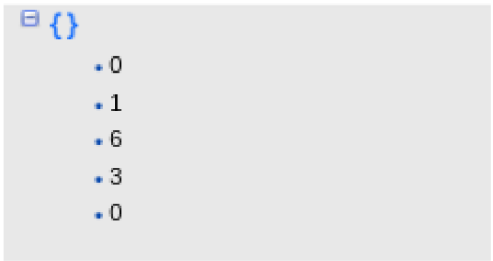

Figure 21. Generated scores109.json result from AgrarianSpider Algorithm. Source:[author]

Also, the Spider examined its own resource utilization and provided a usage report in the 'usage109.json' file. Each Spider run's elapsed time, CPU consumption, memory usage, and network usage are detailed in the usage report, as show in the figure 22.

elapsed_time	cpu_usage	mem_usage	net_sent	net_recv
0.020832538604736328	15.300000000000004	-188416	0	0
0.016890764236450195	31.10000000000001	0	0	0
0.00917196273803711	8.399999999999999	0	0	0
0.0074176788330078125	-16.1	0	0	0
0.1707780361175537	29.300000000000004	708608	0	0
0.008440017700195312	9.899999999999999	0	0	0
0.15720319747924805	18.200000000000003	-172032	0	0
0.11511802673339844	12.599999999999994	0	0	0
0.1526474952697754	22.300000000000004	516096	0	0
0.14429879188537598	17.6	0	0	60
0.1628563404083252	23.60000000000001	155648	0	0
0.29676008224487305	39.3	4468736	211	66
0.14197850227355957	17.699999999999996	-344064	0	0
0.31242966651916504	52.7	1630208	0	0
0.1325206756591797	4.899999999999999	409600	235	74
0.12984609603881836	3.799999999999997	774144	0	0

Figure 22. Generated articles00109.json result from AgrarianSpider Algorithm. Source:[author]

5.2 Discussion

The discussion section of this thesis presents a summary of the proposed web crawling algorithm and its potential use in retrieving articles related to a specific topic. It highlights the technology recommendations and emphasizes the importance of incorporating ethical practices in the data retrieval process.

The Spider algorithm utilised in this study has a number of distinctive characteristics when compared to the previously described related work. First, it concentrates on gathering pertinent information from news websites, particularly those pertaining to agriculture and animals. Its granularity enables more targeted data retrieval and analysis, making it very beneficial for agricultural academics and practitioners.

Second, in order to handle and evaluate text data, the Spider algorithm employs machine learning techniques such as natural language processing. This enables the algorithm to discover and extract useful information from vast volumes of unstructured text input, hence making it more efficient and effective than conventional methods such as human data extraction.

Thirdly, the Spider algorithm focuses a heavy emphasis on the quality and precision of the data. The use of data cleaning techniques and data validation tests ensures the reliability and validity of the obtained data, hence enhancing the reliability of the analysis.

The suggested method stresses the significance of ethical data retrieval procedures. To ensure the trustworthiness and authenticity of the returned data, ethical issues such as respecting the privacy of website owners and adhering to any data protection regulations are essential. By including ethical norms into the data retrieval process, the algorithm is able to generate reliable and accurate findings.

Overall, the Spider algorithm reported in this paper provides a novel and efficient method for retrieving and analysing data from agricultural and livestock-related news websites. Its emphasis on data quality, efficiency, and ethical considerations renders it a significant resource for agricultural researchers and practitioners.

6 Conclusion

In conclusion, this thesis aimed to design and implement an algorithm that crawls web articles focusing on the agrarian sector. The research consisted of a comprehensive literature review on web crawling algorithms and technologies, followed by the practical implementation of the proposed algorithm.

The theoretical part of the work focused on exploring the Internet, World Wide Web, HTML5, and web crawling algorithms, with a particular emphasis on Focused Crawler and Incremental Crawler. Additionally, the chapter presented a comparative analysis of web crawlers and crawling algorithms and highlighted some of the challenges associated with web crawling.

The partial objectives of the thesis were to create a review of similar solutions, select a suitable technology for implementing the algorithm, and validate the algorithm in the context of English-language websites. These objectives were achieved through a rigorous research methodology that involved studying and analyzing professional information sources, implementing the most appropriate technology for the algorithm, and validating the algorithm's efficiency in retrieving articles related to the agrarian sector.

The proposed algorithm was designed to extract information such as category, headline, URL, thumbnail, and summary from web pages. It also included the ability to follow links to related pages and crawl through multiple levels of the website dynamically. The algorithm faced some challenges, including dealing with dynamic content, duplicate content, errors and exceptions, and managing large amounts of data. However, we identified these challenges and addressed them through appropriate methods, such as using dynamic content crawling techniques.

Finally, while the author cannot claim to have achieved the main objective of the thesis, which is to crawl web articles focusing on the agrarian sector successfully, the author believe that his efforts have contributed significantly to the field of web crawling algorithms. The findings have the potential to guide future research in this area and improve the quality and accuracy of web crawling algorithms.

In summary, this thesis has provided a comprehensive review of web crawling algorithms and technologies and presented a practical implementation of a proposed algorithm for retrieving articles related to the agrarian sector. We hope that our work will contribute to the advancement of web crawling algorithms and serve as a foundation for further research in this field.

7 References

1. Leiner, B. M., Cerf, V. G., Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch, D. C., Postel, J., Roberts, L. G., & Wolff, S. (1997). Brief History of the Internet. Internet Society. https://www.internetsociety.org/internet/history-internet/brief-history-internet/
2. Mozilla. (n.d.). How does the Internet work? Retrieved March 2, 2023, from https://developer.mozilla.org/en-US/docs/Learn/Common_questions/How_does_the_Internet_work
3. Mozilla. (n.d.). World Wide Web. Retrieved March 2, 2023, from https://developer.mozilla.org/en-US/docs/Glossary/World_Wide_Web
4. Mozilla. (n.d.). How the web works. Retrieved March 2, 2023, from https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works
5. Mozilla. (n.d.). HTML basics. Retrieved March 2, 2023, from https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics
6. W3C. (2011, April 5). HTML5: A vocabulary and associated APIs for HTML and XHTML - Introduction. Retrieved March 3, 2023, from https://www.w3.org/TR/2011/WD-html5-20110405/introduction.html
7. HTML Semantic Elements. (n.d.). In W3Schools Online Web Tutorials. Retrieved March 3, 2023, from https://www.w3schools.com/html/html5_semantic_elements.asp
8. Mozilla. (n.d.). What are hyperlinks? Retrieved March 2, 2023, from https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_are_hyperlinks
9. Mozilla. (n.d.). URL. Retrieved March 2, 2023, from https://developer.mozilla.org/en-US/docs/Glossary/URL
10. Mozilla. (n.d.). What is a URL? Retrieved March 2, 2023, from https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL
11. Lawrence, S., & Giles, C. L. (1998). Searching the World Wide Web. Science, 280(5360), 98-100.
12. Pavalam, S. M., Jawahar, M., Felix K Akorli, & S. V. Kashmir Raja. (2011). Web Crawler in Mobile Systems. In Proceedings of International Conference on Machine Learning (ICMLC 011) (Vol. , pp.).
13. Castillo, C., Marin, M., & Rodriguez, A. (2004). Scheduling Algorithms for Web Crawling. In Proceedings of WebMedia and LA-Web.
14. Chakrabarti, S., van den Berg, M. & Dom, B., 1999a Focused crawling: a new approach to topicspecific Web resource discovery. In Proceeding of the 8th International conference on World Wide Web, Toronto, Canada, pp.1623.
15. http://en.wikipedia.org/wiki/Distributed_web_crawling
16. Pavalam S M, S V Kashmir Raja, Felix K Akorli and Jawahar. A Survey of Web Crawler Algorithms, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 6, No 1, November 2011 ISSN (Online): 1694-0814.
17. Yongbin Qin and Daoyun Xu “A Balanced Rank Algorithm Based on PageRank and Page Belief recommendation”
18. Alexander Shen —Algorithms and Programming: Problems and solutions‖ Second edition Springer 2010, Pg 135

19. S. N. Sivanandam, S. N. Deepa “Introduction to Genetic Algorithms” Springer, 2008, pg 20
20. Shaojie Qiao, Tianrui Li, Hong Li and Yan Zhu, Jing Peng, Jiangtao Qiu “SimRank: A Page Rank Approach based on similarity measure” 2010 IEEE.
21. TIAN Chong “A Kind of Algorithm For Page Ranking Based on Classified Tree In Search Engine” Proc International Conference on Computer Application and System Modeling (ICCASM 2010).
22. Pann Yu Mon, Chew Yew Choong, Yoshiki Mikami, “Language Specific Crawler for Myanmar Pages”. In Proceedings of the 11th International Conference on Humans and Computers (HC 2008), Nagaoka, Japan, November 2008
23. Nomura, S., Oyama, S., Hayamizu, T.: Analysis and Improvement of HITS Algorithm for Detecting Web Communities. Journal of Systems and Computers 35(13) (2004).
24. Wenxian Wang, Xingshu Chen, Yongbin Zou, Haizhou Wang, Zongkun Dai “A Focused Crawler Based on Naive Bayes Classifier” Third International Symposium on Intelligent Information Technology and Security Informatics, 2010.
25. Python Software Foundation. (2021). Python 3.10.0 documentation. Retrieved from https://docs.python.org/3/
26. Scrapy. (2021). Documentation. Retrieved from https://docs.scrapy.org/en/latest/
27. Visual Studio Code. (2021). Features. Retrieved from https://code.visualstudio.com/docs/editor/features

8 List of pictures, tables, graphs and abbreviations

8.1 List of pictures

Figure 1. simple connection between two computers. Source:[2]
Figure 2. Computer network connection between more computers. Source:[2]
Figure 3. Router with cabled-connected computers. Source:[2]
Figure 4. Two Routers connection with a cable. Source:[2]
Figure 5. Four Routers connection via cable. Source:[2]
Figure 6. Router connection with a modem. Source:[2]
Figure 7. Connection between ISP and Final user. Source:[2]
Figure 8. Locate a Computer using its domain name and IP address. Source:[2]
Figure 9. simple formation of the Internet and the World. Source:[2]
Figure 10. simple connection between two computers. Source:[2]
Figure 11. HTML <section> Element. Source:[author]
Figure 12. HTML <article> Element. Source:[author]
Figure 13. HTML <nav> Element. Source:[author]
Figure 14. HTML <aside> Element. Source:[author]
Figure 15. HTML <figure> Element. Source:[author]
Figure 16. Architecture of a Basic Crawler. Source:[author]
Figure 17. Architecture of a Breath-First Search Algorithm. Source:[25]
Figure 18. Architecture Component Diagram of theAgrarianSpider Algorithm. Source: [author]
Figure 19. Architecture Component Diagram of the AgrarianSpider Algorithm 2. Source: [author]
Figure 20. Generated articles00109.json result from AgrarianSpider Algorithm. Source: [author]
Figure 21. Generated scores109.json result from AgrarianSpider Algorithm. Source: [author]
Figure 22. Generated articles00109.json result from AgrarianSpider Algorithm. Source: [author]
Figure 23. Generated articles00109.json result from AgrarianSpider Algorithm. Source: [author]

8.2 List of tables

Table 1 Advantages and Disadvantages of Crawler

Table 2 Pros and Cons of Crawling Algorithms

8.3 List of Source Code

Source code 1 AgrarianSpider – partital component 1
Source code 2 AgrarianSpider – partital component 2
Source code 3 AgrarianSpider – partital component 2
Source code 4 AgrarianSpider – parse method 1
Source code 5 AgrarianSpider – parse method 2
Source code 6 AgrarianSpider – process_item method
Source code 7 AgrarianSpider – is_duplicate method
Source code 8 AgrarianSpider – score_article method -parital component
Source code 9 AgrarianSpider – close method -parital component 1
Source code 9 AgrarianSpider – close method -parital component 2