Web Data Mining with Python

Discover and extract information from the web using Python

Dr. Ranjana Rajnish
Dr. Meenakshi Srivastava

bpb

Web Data Mining with Python

First published: 2023

Published by BPB Online

WeWork

119 Marylebone Road

London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-5551-366-3

www.bpbonline.com

Dedicated to

Dear God,

and

Our Parents

About the Authors

Dr. Ranjana Rajnish, is an Assistant Professor at Amity Institute of Information Technology at Amity University Uttar Pradesh, Lucknow. Dr. Ranjana possesses more than 25 years of experience in academics/research. She has been engaged with institutions like U.P. Technical University and Amity University in roles ranging from a faculty in computer science to Academic Head. She has done her Ph.D. in Computer Science and her area of teaching and research includes Programming Languages such as C, Python, Borne Shell, Software Engineering, Opinion Mining/Sentiment Analysis, and Healthcare. She has more than 40 publications in National and International conferences of repute.

Dr. Meenakshi Srivastava, is working as Assistant Professor at Amity Institute of Information Technology, Amity University, Uttar Pradesh, Lucknow Campus since 2005. She has done her Ph.D. in Engineering. Her area of teaching and research includes Web Mining, Image Processing, Business Intelligence, and Bioinformatics. She has more than 18 years of teaching experience, where she has taught various programming languages like C, C++, C#, JAVA, Python and R. She has more than 40 national and international publications.

About the Reviewer

Dr. Deepak Singh completed his doctorate from Indian Institute of Technology, Kanpur and has over 13 years of experience in the field of academics and IT industry. His core areas of expertise include real-life applications of artificial intelligence and IoT based systems. Apart from teaching in prominent institutions in Lucknow, he is also consultant of several IT firms. He has also advised UP government on IT implementation in Public Libraries.

He has played a key role in several AI based research projects, which include Digital Transformation of several prominent educational institutes of Lucknow, two projects funded by Government of India and several other projects related to AI-ML implementation. He has imparted training to several senior level executives of IT companies in different parts of world including Singapore, Japan and China.

Acknowledgements

Acknowledgment is the most beautiful part of any project as it gives an opportunity to express gratitude to people who have either helped or have been constantly supported throughout the project.

Writing a book is no less than a project. During this journey of writing a book, many people helped in small or big ways.

Our list of acknowledgements begins with the almighty God.

We are thankful to Ms. Prerna Mishra, who contributed to the writing of chapter 5, and our students, Mr. Ayush Kumar Rathore, Mr. Anant Gupta, and Mr. Apurva, who helped in executing a few scripts and figures in this book.

We are grateful to Dr. Deepak Singh, whose valuable comments have helped us to improve the contents of the book. He gave us support throughout the process of book writing.

Our gratitude also goes to the team, Ms. Surbhi and Ms. Suman, at BPB Publication, for being constant support to finish the book and allowing us to publish the book. We are also thankful to other editors of BPB, whose input helped in improvising the book.

Finally, we would like to thank our family, especially the kids; without their support, it wouldn't have been possible to complete this book.

Happy Learning.

Preface

The idea of writing this book came when we were doing some research using Web Mining. We found it very difficult to get one single book, that helped us understand the fundamentals along with its implementations. Also, most of the books assumed that the readers have prior knowledge of Python. This book starts with the fundamentals of web mining and Python so that even beginners can follow the subject.

Before inviting the readers to go through the content of this book, we would like to take an opportunity in this preface to present the readers with an overall roadmap of the book.

Nowadays it is very common that whenever any query comes to our mind, we tend to search for it on the web using any of the search engines (like google, yahoo, etc.). With over 560 million internet users in India, just behind China, India holds number two rank in internet usage. This makes us understand the volume of people who are accessing the internet for various purposes like E-commerce, social media, e-learning etc.

Enterprises are moving rapidly towards monetizing Digital Assets.

E-commerce came into the picture in May 1989, when Sequoia Data Corp. introduced Compumarket - the first internet-based system for e-commerce. This system supported sellers to post items for sale and then the buyers could search the database for the item of their interest and make purchases. Compumarket supported the use of credit cards also. Since then, E-commerce has transformed the global retail industry and business organizations have started demanding automated intelligent solutions like anticipating customer expectations and requirements. With this information they not only improve the customer satisfaction level but also help the seller in maintaining adequate stock inventory and provide input to augment sales volume. Web Mining plays a crucial role in prediction of these customer expectations and required inventory. This information can be used by businesses for effective decision-making. Similarly, data from various social media sites could be analyzed and could be used by several industries for business development, social science research, health services, and educational purposes. The vast application of web mining was the prime source of motivation behind writing this book.

We as authors, were willing to have a one-window solution for writing web mining-based applications. The book will provide a complete overview of theoretical and mathematical modeling of various algorithms used in web mining as well as python-based solutions for implementing them.

The proposed book on “Web mining using Python” is based on the Concept and Applications of mining information from the web, which is the most sought area in

the field of Data Science.

As Data Science is the fastest growing job on LinkedIn and is predicted to create 11.5 million jobs by 2026, so the job seekers with this skill set have lot of opportunities. The purpose of this book is to provide sufficient knowledge in the field of web mining to the reader who aspires to be a Data Scientist.

The book is divided into three broad sections, Introduction & Concepts, Methodologies and Applications.

Chapter 1: Web Mining – An Introduction, deals with the introduction, evolution and basic concepts of the Web Mining.

Chapter 2: Web Mining Taxonomy, covers its taxonomy, Web content mining, Web structure mining, Web usage mining, and other important concepts.

Chapter 3: Prominent Applications with Web Mining, discusses Prominent Applications with Web mining, personalised customer applications, Web search, Web wide tracking and Process mining.

Chapter 4: Python Fundamentals, elaborates various applications with mining methodologies and fundamentals of Python Programming Language. It will also cover, Basics of Python, Basic HTML tags and Basics of Python Libraries.

Chapter 5: Web Scraping, covers the uses of Web scraping, working of Web scraping, Python modules and legality of Web scraping, Data Extraction and Preprocessing.

Chapter 6: Web Opinion Mining, talks about the Concepts of Opinion Mining, Data Processing and Tokenization, and Feature extraction.

Chapter 7: Web Structure Mining, focuses on the concept and types of Web Structured Mining. It also covers Web Graph Mining, Deep Web Mining, Web Search and Hyperlinks.

Chapter 8: Social Network Analysis in Python, explains Introduction to Social Network Analysis, Creating Symmetric and Asymmetric Network, Network Connectivity.

Chapter 9: Web Usage Mining, covers the sources and types of Data, Key Elements of Web Usage Data Pre-processing, Data Modeling, Discovery and Analysis of Pattern.

Web mining is a vast and very active area of research, we have tried to explain all the concepts without making the book too voluminous. We have made an effort to keep the book simple and understandable, and we will be happy to receive feedback from our readers. We hope this book will help readers in learning the concepts of web mining using python.

Dr. Ranjana Rajnish

Dr. Meenakshi Srivastava

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/av77i9u

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Web-Data-Mining-with-Python**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Table of Contents

CHAPTER 1

Web Mining—An Introduction

Introduction

Web mining is the process of discovering and extracting information from the Web using various mining techniques. This information can be used by businesses for effective decision-making. This chapter introduces you to World Wide Web, the basics of data mining, and Web mining discusses the type of information that can be mined and its applications. It also discusses about how Python can be used in Web mining. This chapter is meant for the beginner-level reader who is a novice in the field of Web mining. The purpose of this chapter is to give a broad introduction so that you can understand the following chapters.

Structure

In this chapter, we will discuss the following topics:

- Introduction to Web mining
- Word Wide Web
- Internet and Web 2.0
- An overview of data mining, modeling, and analysis
- Evolution of Web mining
- Basics of Web mining
- Applications of Web mining
- Web mining and Python
- Conclusion
- Questions and exercises

Objectives

After studying this chapter, you will be able to have an introduction to Web mining, the evolution of the Web, and basic concepts of Web mining, which will be followed by how Web mining is different from data mining. You will also be able to understand why Python is helpful in Web mining and what are relevant steps needed for mining information.

Introduction to Web mining

Web mining is the process of discovering and extracting information from the Web using various mining techniques. This information can be used by businesses for effective decision-making.

In earlier days, data was stored in databases and was in a structured form; thus, any information could be fetched by writing queries on those databases. Information dissemination was then in the form of reports generated from the data stored in the database. Now, **World Wide Web** (**WWW**) has become the most popular method to disseminate information; thus, there is an information overload on the Web. The Web has changed the way how we perceive data, whereas Web 3.0 is characterized by the Web with the database as one of its features. The Web as a database gives us the possibility of exploring the Web as a huge database that is full of information. Using **Knowledge Discovery** (**KD**) processes, meaningful information can be extracted from this huge database having a variety of information such as text, image, video, and multimedia.

Gone are the days when people used to go to the library to read. In case of any query that comes to mind, we tend to search it on the Web using any of the search engines (such as google, yahoo, and so on). With over 560 million internet users in India, just behind China, India holds the number two rank in internet usage. This makes us understand the volume of people who are accessing the internet for various purposes. With so much data available across the internet, we need to convert it to relevant information that could be used for some meaningful application. To take full advantage, data retrieval is not sufficient, and we need a methodology that helps us to extract data from www and converts it into meaningful information.

Web Mining is the process of mining or extracting meaningful information from the Web. Other two commonly used definitions of Web Mining are as follows:

"Web mining is the use of data mining techniques to automatically discover and extract information from Web documents/services" (Etzioni, 1996, CACM 39(11)).

"Web mining aims to discovery useful information or knowledge from the Web hyperlink structure, page content and usage data."(Bing LIU 2007, Web Data Mining, Springer).

World Wide Web

World Wide Web, commonly referred as www or internet, had its modest beginning at CERN, an international scientific organization in Geneva, Switzerland, in 1989 by Sir Berners Lee, a British Scientist. According to Lee, sharing information was difficult as you had to log on to different computers for it. He thought to solve this problem and submitted his initial proposal called "*Information Management: A proposal*" in March 1989. Please refer to the following image:

Figure 1.1: *Tim Berners-Lee at CERN (Image: CERN)*
Source: *https://www.britannica.com/topic/World-Wide-Web*

He formalized the proposal and submitted a second version of it in 1990 along with Belgian systems engineer Robert Cailliau. In this proposal, he outlined the concepts related to the Web and called it a "hypertext project" called "WorldWideWeb". They proposed that the Web will consist of "hypertext documents" that could be viewed by browsers. Tim Berners-Lee developed the first version with a Web server and a browser running, which demonstrated the ideas he presented in the proposal. The Web address of the first website was **info.cern.ch**, which was hosted on a NeXT computer at CERN. The website contained information about the WWW project, including all details related to it. The address of the first Web page was **http://info.cern.ch/hypertext/WWW/TheProject.html**. To ensure that the machine that was used as a "Web server" was not switched off accidentally, it was explicitly written using red ink that "This machine is a server. DO NOT POWER IT DOWN".

Initially, the Web was conceived and developed for automated information sharing between scientists in universities and institutes around the world. The following figure is a screenshot showing the NeXT World Wide Web browser:

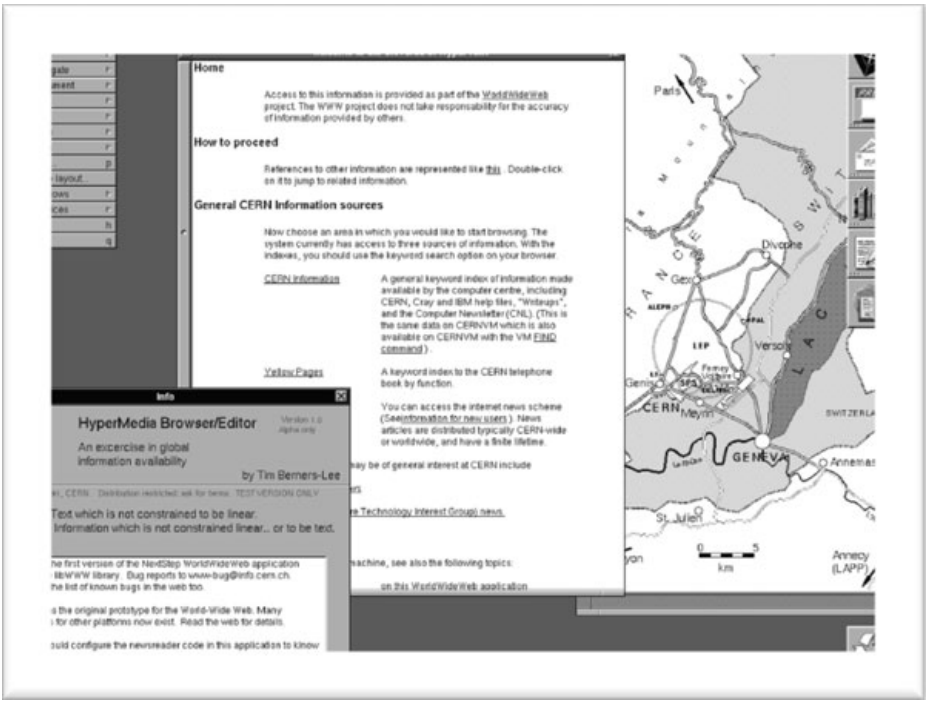


Figure 1.2: *A screenshot showing the NeXT World Wide Web browser created by Tim Berners-Lee (Image: CERN)*

The website allowed easy access to existing information useful to CERN scientists. It provided a search facility on keyword basics as there was no search engine at those times. This project had limited functionality at the beginning, where only a few users had access to the NeXT computer platform (which was used for the server). In March 1991, this was made available to all colleagues using CERN computers. In August 1991, Berners announced the www version of the Web on Internet newsgroups, and the project spread around the world.

The European Commission joined hands with CERN, followed by which CERN made the source code of WorldWideWeb freely available. By late 1993, there were more than 500 known Web servers running, and the WWW accounted for 1% of internet traffic (others were e-mail, remote access, and file transfer).

Tim had defined three basic building blocks of the internet as HTML, URI, and HTTP, which remain the foundations of today's Web.

Hyper Text Markup Language (HTML) is used as markup(formatting) language.

Uniform Resource Identifier (URI), also known as Uniform Resource Locator (URL) used as a unique address to locate each resource on the Web.

Hypertext Transfer Protocol (**HTTP**) is the protocol that helps retrieve linked resources from the Web.

Its popularity increased manifolds when software giant Microsoft Corporation lent its support to internet applications on personal computers and developed its own

browser, **Internet Explorer** (**IE**), which was initially based on Mosaic in 1995. Microsoft integrated IE into the Windows operating system in 1996; thus, IE became the most popular browser.

Evolution of the World Wide Web

World Wide Web has evolved tremendously from the time it was developed till now. Each period of its evolution has added a lot of value to it and can be categorically distinguished with distinct concepts associated with it. In this section, we have given a brief of how the Web has evolved.

By the end of 1994, around 10,000 servers were being used, of which 2,000 were commercial. The Web was being used by over 10 million users by then, and internet traffic increased immensely. Technology was continuously explored to cater to other needs, such as security tools, e-commerce, and applications.

Initially, the basic version, that is, **Web 1.0 (1989)**, was designed to publish information that could be read by all. This era was characterized by the hosting of informative websites that published corporate information such as organizational information, brochures, and so on to aid in businesses. So, we can say it was a collection of a huge number of documents that could be read across the World Wide Web. The main objective of this era was to create a common information place from where information sharing could take place. **This era was read-only Web, which consisted of static HTML pages.**

Note: Web 1.0 was designed for information sharing and allowed only to publish information on the website. The user could only read information.

In 2004, Web 2.0 evolved and was known to be a **people-centric Web, participative Web, and social Web**. It was used as a collaboration platform as it became bidirectional, where read-write operations could be done, making it interactive. In this version, the Web technologies used facilitated information sharing, interoperability, user-centered designs, and collaborations. This was the time when services, such as wikis, blogs, YouTube, Facebook, LinkedIn, Wikipedia, and so on, were developed. So, **this era is characterized by read and write both** and in addition to documents, and even the users got connected.

Note: Web 2.0 is characterized by a people-centric Web, participative Web, and social Web where users can read and write on the Web.

Web 3.0, the third-generation Web, was conceived as the Web that helped in more effective **discovery, automation**, and **integration** as it combined human and artificial intelligence to provide more relevant information. Web 3.0 emphasizes on analysis, processing, and generating new ideas based on the information available across the Web. Web 3.0 coined a new concept of transforming the Web into the database; thus, making it more useful for technologies such as Artificial Intelligence, 3D graphics, Connectivity, Ubiquity, and Semantic Web. It is also

known as the semantic Web and was conceptualized by Tim Berners-Lee, the inventor of Web 1.0. The semantic Web emphasizes upon making the Web readable by machines and responding to complex queries raised by humans based on their meaning. According to W3C, "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries".

Note: Web 3.0 is characterized by technologies such as Artificial Intelligence, 3D graphics, Connectivity, Ubiquity, and Semantic Web.

Web 4.0 is more revolutionary and is based on wireless communication (mobile or computers), where people can be connected to objects. For example, GPS-based cars help the driver to navigate the shortest route. This generation is termed as an "Intelligent Web" and will be seen between 2020 and 2030. In this generation, computers have varied roles, from personal assistants to virtual realities. The Web will have intelligence like humans, and highly intelligent interactions between human and machine will occur. All household appliances may be connected to the Web, and work will be done on brain implants.

Note: Web 4.0 is seen as the "Mobile Web" or the "Intelligent Web".

Web 5.0 is a futuristic Web with a lot of research going on. It is projected to be "The Telepathic and Emotional Web" and should come by 2030.

Internet and Web 2.0

Internet and the emergence of Web 2.0, known as the people-centric Web, participative Web, and social Web, provided new ways of how information on the internet can be harnessed and used for the benefit of society. This was the time when a fundamental shift of how we use the internet happened. Earlier, the internet was used as a tool; in Web 2.0 internet became part of our life as it became a social Web from a static Web. In this era, we have not only increased our usage data but also increased internet usage time. Websites became more interactive, and new technologies allowed websites to interact with a Web browser without human intervention.

Use of various smart mediums such as smartphones, Tablets, Laptops, MP3 players, and various tools such as Search Engines (for example, Google and Yahoo); video and photo sharing tools (for example, YouTube and Instagram); and social networking mediums (for example, Facebook and WhatsApp) internet has become an integral part of our life. A lot of data is being generated through various platforms in the form of text, images, and videos. This led to information overload; thus, it was important to extract meaningful and significant data from the large volume of data available on the Web. This led to the emergence of technologies like Web mining for informational retrieval.

An overview of data mining, modeling, and analysis

The use of the internet to develop online business applications in all fields and the automatic generation of data through various sources across the internet led to extremely large repositories of data. Stores like Walmart and Big Bazar have thousands of stores having millions of transactions per day. A lot of technical innovations took place to manage how this huge data can be stored. Database management technologies were developing fast to manage data, but methodologies used for retrieval and analysis were trivial. It was only when companies started realizing that in this huge raw data, there is a lot to be explored that computer scientists started working on how this hidden information can be explored. The huge amount of data had many hidden facts or patterns that could be explored to help in having a better decision support system to make more effective decisions. The data contained a lot of knowledge about the number of aspects related to the business that could be harnessed to have effective and efficient decision-making. **This extraction of knowledge from the databases or datasets is known as Data Mining or Knowledge Discovery in Databases (KDD).**

What is Data Mining?

According to Gartner, "*Data Mining is the process of discovering meaningful new correlations, patterns and trends by shifting through large amounts of data stored in repositories, using pattern recognition technologies as well as statistical and mathematical techniques.*"

Understanding the potential of Data Mining, a lot of technologies were developed that could help in analyzing this huge data, often termed as Big Data. There was a big shift in the focus of the market from products to customers, and the trend shifted to provide personalized transactions. The technology used to capture data also shifted from manual to automated using Bar Codes, POS devices, and so on. Database management technologies were initially used for efficient storage, retrieval, and manipulation of data, but with this new requirement of mining of information, a lot of algorithms were developed to mine this information. That was the time when Machine Learning also started evolving, and with the combination of data mining techniques and machine learning algorithms, there came a revolution in the mining field.

Note: Big data is a huge amount of data that is characterized by volume, velocity, and veracity. This could be analyzed computationally to find out the hidden pattern, trends, and associations.

Data mining uses concepts of database technology, statistics, machine learning, visualization, and clustering, and please refer to the figure 1.3:

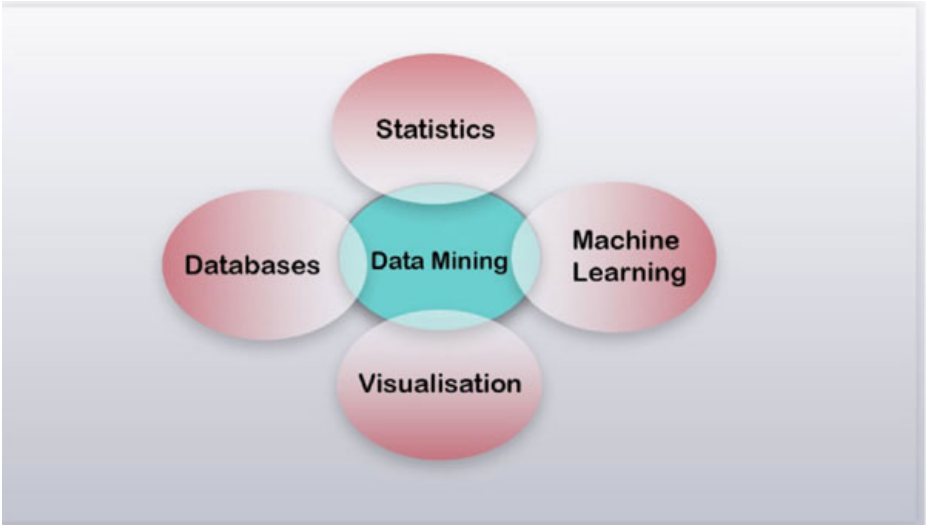


Figure 1.3: *Concepts used in data mining*

What is data mining, and what is not? Many users are having confusion about how data mining is different from a regular database search. Let us see with the help of an example:

Searching for a phone number in a telephone directory **is not Data Mining.**

Searching for students who have scored marks more than 75% **is not Data Mining.**

Searching for a string like "Data Mining" using the search engine **in not Data Mining**.

Analyzing the customer's buying pattern based on his past purchase pattern **is Data Mining.**

Making personalized recommendations to online shoppers, YouTube viewers or Saavn (an online music streaming service), or OTT platforms such as Amazon Prime, Disney Hot Star, and so on **is Data Mining.**

Data modeling/mining process

Almost all the verticals of business-like marketing, fast-moving consumer goods (FMCG) and aerospace are taking advantage of data mining; thus, many standard data models have been developed. A standard data mining process consists of the following steps:

1. **Selection**: It is the process in which the data relevant for analysis is retrieved from data sources.
2. **Pre-processing**: It is important to process the data as it is vital to have good quality data to make it useful. Data is said to be useful if it possesses attributes such as accuracy, consistency, and timeliness. Thus, pre-processing becomes a very critical step in data mining. During this process, the major steps involved are as follows:

a. Data cleansing, to fill in the incomplete data or remove noisy data.

b. Data integration, to combine data from multiple heterogeneous data sources such as files, databases, and cubes.

c. Data reduction to obtain relevant data for analysis while maintaining its integrity.

3. **Transformation**: This process transforms data into data suitable form for data mining so that the mining process is more efficient and it is easier to mine the patterns.
4. **Data Mining**: This process extracts patterns using intelligent algorithms. Patterns are structured using clustering and classification techniques.
5. **Interpretation**: This step uses methods like data summarization and visualization of the patterns.

Please refer to the following figure:

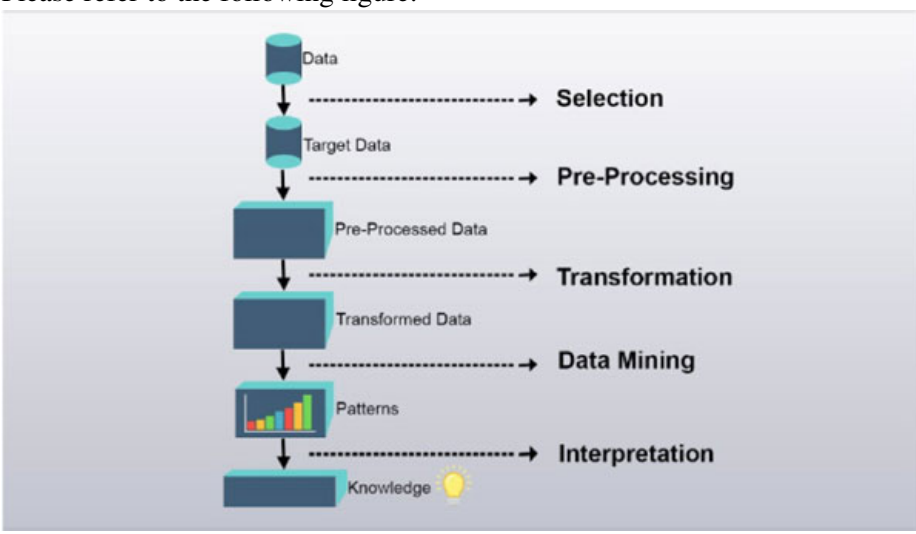


Figure 1.4: *Steps used in data mining*

Basics of Web mining

World Wide Web, often referred as the Web, has become the most popular medium of disseminating information. The information is huge, diverse, and dynamic, which raises the issues of scalability, temporal issues, and issues related to multimedia content. This huge source of data can be used for finding relevant information, creating knowledge from the information available on the Web, personalization of information, or learning about consumers or individual users.

Web mining helps us in automatically discovering and extracting information from Web resources/ pages/documents by using various data mining techniques. Some of the examples of Web resources are electronic newsletters, text content of Web pages from websites (removing HTML tags), electronic newsgroups, and so on.

Web data is mostly unstructured data, such as free text on Web pages, or semi-structured data, such as HTML documents and data in tables.

In the last several years, most of the government data has been ported onto the Web, and almost all the companies have their own websites or Web-based ERP systems that continuously generate data. Also, digital libraries are accessible from the Web, E-commerce sites, and other companies are doing their business electronically on the Web. Companies, employees, customers as well as their business partners are accessing all data from the Web-based interfaces. As a result, we see that the Web consists of a variety of data such as textual data, images, audio data, video data, multimedia data, hyperlinks, and metadata.

This information can be seen from two sides, one is from the user's point of view, and the other is from the information provider's point of view.

- User's perspective: browsing or searching for relevant information on the Web.
- Information provider's perspective: providing relevant information to the user.

Information provider's problem is to find out What do the customers want? How effectively Web data can be used to market products or services to customers? How to find the pattern in user buying to make more sales?

Web mining can provide an answer to all these questions.

Categories of Web mining

Web mining is broadly categorized as the mining of W**eb Contents**, mining of W**eb Structure**, and mining of **Web Usage Data**.

- Web content mining: is extracting knowledge from the content of the Web.
- Web structure mining: is discovering the model underlying the link structures of the Web.
- Web usage mining: is discovering user's navigation pattern and predicting user's behavior.

We will see Web content mining, Web structure mining, and Web usage mining in detail in *Chapter 2, Web Mining Taxonomy*. The following figure shows the categories of Web data mining:

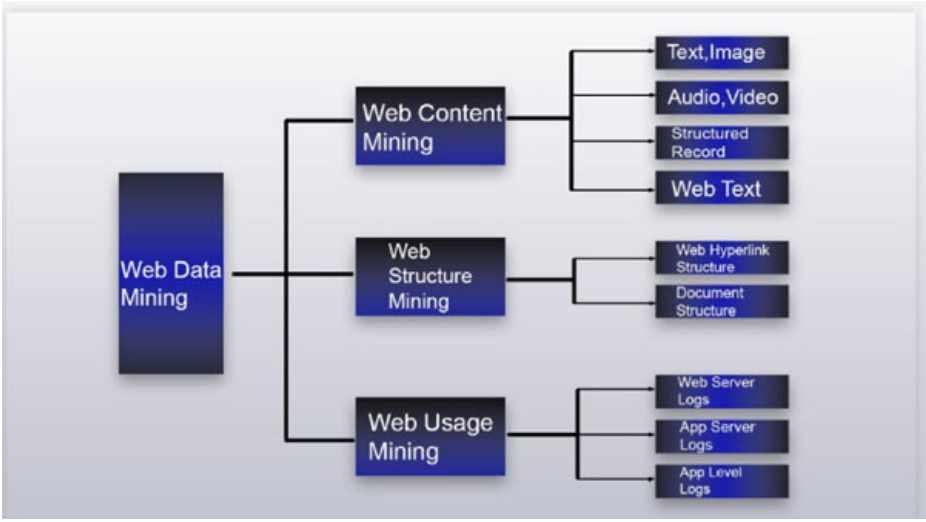


Figure 1.5: *Categories of Web data mining*

The Web mining task is decomposed into the following major steps:

- **Resource discovery/data collection:** This step performs the task of retrieving the Web documents such as electronic newsletters, text content of Web pages from websites (removing HTML tags), electronic newsgroups, and so on. These documents are then used for information extraction.
- **Information Extraction:** In this step, specific information from Web resources is retrieved and pre-processed.
- **Pattern Discovery:** In this step, the discovery and identification of general patterns in the Web pages from a single website or multiple websites are done.
- **Pattern Analysis:** In this step, analysis and interpretation of patterns in the mined data is done using various visualization tools; the steps are showcased in the following figure:

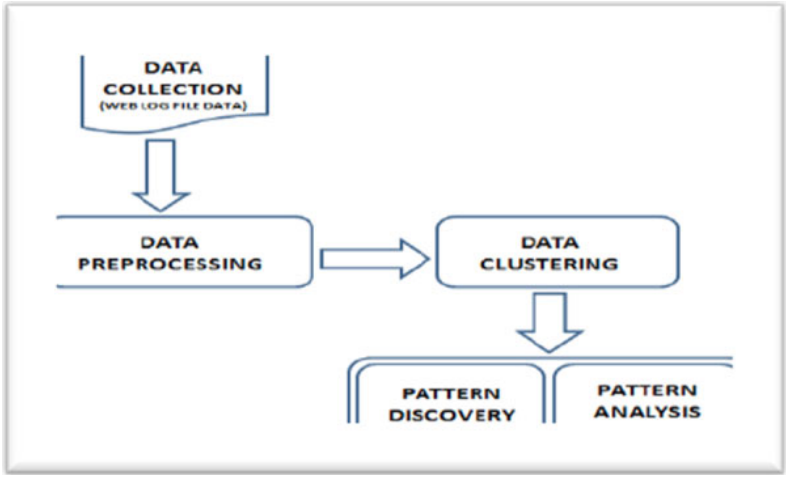


Figure 1.6: *Steps of Web Mining*

- **Resource discovery:** This step performs the task of retrieving the Web documents from which information is to be extracted.
- **Information Extraction/Retrieval**: This process performs a series of tasks on the information retrieved from Web sources and aims at transforming the original data before it is ready for mining. After information retrieval, it performs data pre-processing which primarily removes outliers and[1] tokenization, lowercasing, stop words, and so on. It is only after this step the data is ready to be mined for the hidden pattern using data mining techniques.
- **Pattern discovery**: Web mining can be viewed as an extension of **Knowledge Discovery From Databases** (**KDD**). This step uses various data mining techniques for the actual process of discovery of potentially useful information from the Web. Pattern discovery aims at discovering interesting patterns, which include a periodic or abnormal pattern from the temporal data.
- **Pattern analysis:** The step aims at finding out the relationship between the patterns mined. This is often termed as data visualization.

Several tools, such as R, Weka, Python, Rapid Miner, and Oracle Data Mining, can be used for pre-processing and pattern discovery. Pattern analysis can be done with the help of visualization tools such as Tableau, R, Weka, Google Charts, and so on.

Note:

1. Tokenization is splitting the sentences into words,
2. lowercasing is converting all characters to lowercase,
3. stop word is removing all words such as a, an, the, and so on,
4. temporal data is the historical data.

Difference between data mining and Web mining

Both these terms, that is, data mining and Web mining, create a lot of confusion, and few people use them interchangeably. Though they may seem as same, there are some fundamental differences between them. The difference between data mining and Web mining is shown in the next table:

CHAPTER 2

Web Mining Taxonomy

Introduction

The internet has evolved into a global space in which all kinds of information are available. Thus, it has become a prime source of information of all types. The information can be in the form of text, image, audio, video, and animation. Web mining provides techniques that can be used to extract information from semi-structured and unstructured Web pages. Web mining itself is a broad area, so to understand it better, it has been classified into three distinct areas, that is, Web content mining, Web structure mining, and Web usage mining. This chapter introduces the reader to these types, followed by other related key concepts. Also, the chapter discusses other related aspects such as Ranking Metrics, Page Rank, Hubs and Authorities, Web Robots, Information Scent, User Profile, and Online Bibliometrics.

Structure

In this chapter, we will discuss the following topics:

- Introduction to Web mining
- Web content mining
- Web structure mining
- Web usage mining
- Key concepts
- Ranking metrics
- Page rank
- Hubs and authorities
- Web robots
- Information scent
- User profile
- Online bibliometrics

Objective

While going through this chapter, you will learn the concept and taxonomy of Web mining. You will also understand the process of Web mining and ranking metrics.

Introduction to Web mining

Web mining is the process of using data mining techniques and algorithms to extract information directly from the Web. Web mining is performed by retrieving information from Web documents, Web services, Web content, hyperlinks, and server logs. Basically, **Information Retrieval** (**IR**) is a significant area in Web mining where the users procure their needed information from the Web.

The meaning of the term Information Retrieval can be very broad. For example, searching for someone's phone number in the contact list or searching for any previous mail in your inbox are forms of Information Retrieval. However, with reference to the WWW, Information Retrieval may be defined as follows:

Finding material (usually documents) of an unstructured nature (usually text) that satisfies the information need from within large collections (usually on local computer servers or on the internet).

Earlier, Information Retrieval was an activity that was used to refer to the people who were engaged in the profession of searching, such as librarians, paralegals, and other similar professional searchers. However, the scenario has changed, and now hundreds and millions of people are engaged in Information Retrieval every day while using the Web search engine or accessing their e-mails. Information Retrieval is fast becoming the dominant form of information access, overtaking traditional database-style searching.

IR is a science and practice of identifying documents or sub-documents that meet information needs. Usually, IR deals with textual documents in semi-structured (for example, HTML and XML) or unstructured (plain text) format. There are other branches of IR working on multimedia formats such as pictures, audios, and videos, both in raw and structured formats (such as MPEG7). Information needs might include questions to answer, topical descriptions, specifications dealing with time and space, or other such information.

The research domain concerning IR has made a long journey. In earlier times, conventional methodologies involved structuring and storing textual information in various databases to pave the way for well-defined query accesses.

Broadly, the models for the retrieval of information are classified into the following three categories:

- **Algebraic methods:** In this method, the documents are typified as matrices, tuples, or vectors, which get revamped into a one-dimensional similarity aspect deploying algebra. The Vector Space model, the most popular and widely accepted model of IR, is a well-acclaimed example of this.
- **Probabilistic approaches:** In this method, the appositeness of the document is determined by the weight of the words it comprises.
- **Boolean strategies:** In this method, the documents are characterized by laid down terms; the resemblances are obtained using predefined theoretic operations, and this model merely examines the absence or presence of

inquired terms in the document.

Let us consider the Web as a large-scale document collection for which we may apply any of the classical text retrieval techniques. Extracting information from Web documents on the basis of text content present in the document may not result in an effective retrieval as the Web is a hostile environment, where the Web search engines have to deal with many factors like a link to the document, the quality of the document, and so on. The effectiveness of Web Information Retrieval can be improved by enabling the retrieval through the unique features of Web documents and Web structure. The unique feature of the Web documents and the structure of the Web provides new sources of information that enhances the effectiveness of retrieval. While retrieving information from the Web, the content written in a Web document, metadata specified with the documents, graph-based structure of the Web and search behavior of the users, and so on are used by the Web Information Retrieval System.

Web mining itself is a broad area, so to understand it better, it has been classified into the following three distinct areas:

- Web content mining (WCM)
- Web structure mining (WSM)
- Web usage mining (WUM)

These three categories aim at the procedure of knowledge discovery of implicit, previously unknown, and probably useful information from the Web.

Depending on the objective and process of Web mining, it is further classified into sub-categories. The Web mining taxonomy is depicted in *figure 2.1*:

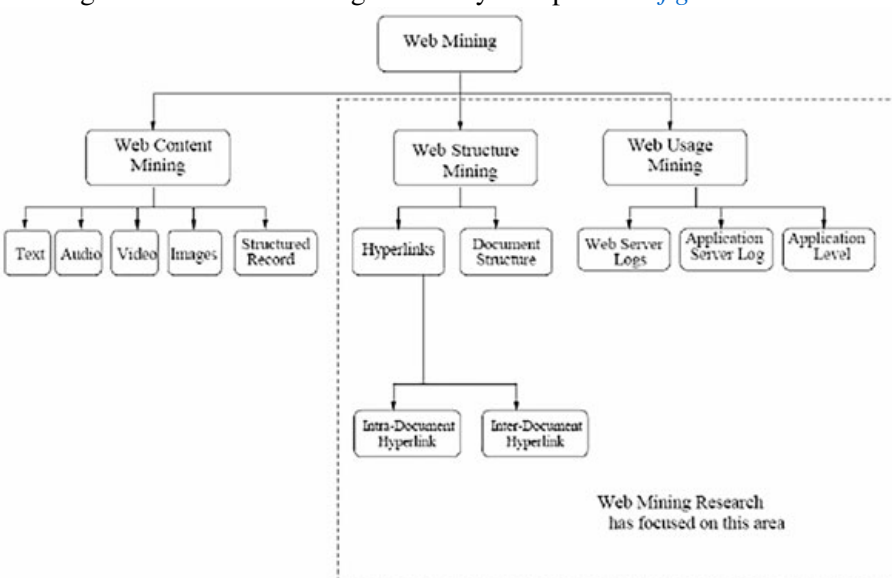


Figure 2.1: *Web mining taxonomy [Web Mining for Business Computing, 2020]*

Web content mining

Web content mining is a process of extracting useful information from the contents of Web documents. Basically, data mining techniques are applied to the content of Web documents so that useful information/pattern can be extracted. *Figure 2.1* depicts the relationship between Web mining categories and their associated common data mining algorithm. Other data mining algorithms can also be used for mining the Web.

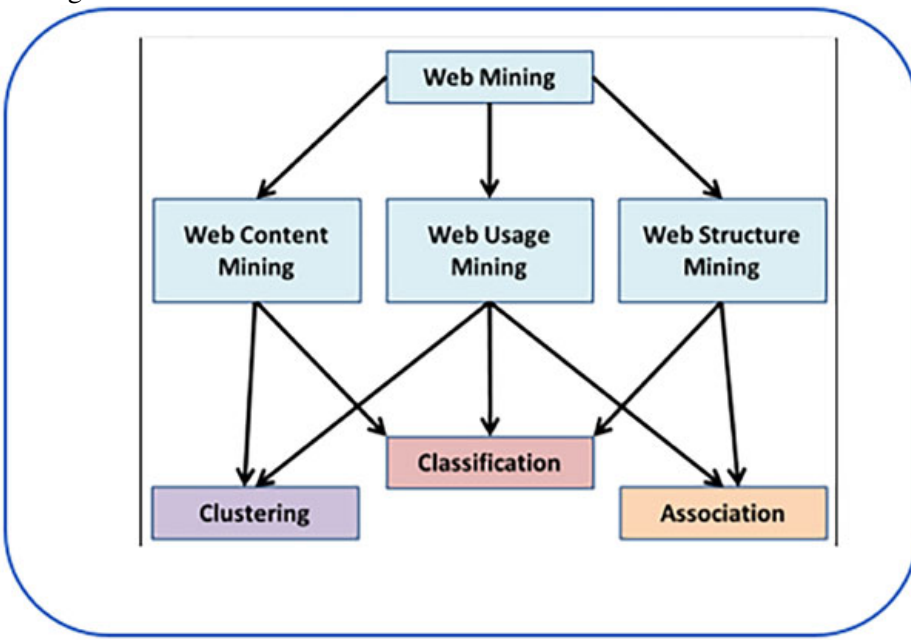


***Figure 2.2**: Association between Web mining and data mining*

Before discussing the process of Web content mining, let us first take a look at the basic applications of Web content mining and how the different data mining algorithms are used in those applications.

Basic application areas of Web content mining

- **Classification of Web Documents:** Suppose you have been given a set of Web documents, and you are required to classify them into various groups. For example, whether the document belongs to the academic group or entertainment group. For such activities, the contents of the Web page will be analyzed, and finally, any suitable data mining technique, such as clustering, may be applied to classify the Web documents.
- **Similarity Search in Web Documents:** We know that on the World Wide Web, Web documents are located on different Web servers. One prominent application of Web content mining is finding similar Web pages across different Web servers. Again, the content of the Web pages will be analyzed

first, and then the clustering techniques will be applied to group the Web documents.

- **Identifying Title/Summary of Web Documents:** Content analysis is applied to the Web documents so that the summary or title of the Web documents is identified.
- **Relevance-based Services**: In data mining, we have recommender systems. **Recommender systems** are an important class of **machine learning** algorithms that offer "relevant" suggestions. We have seen that banner ads are becoming effective techniques for advertisements on the Web. Business organizations pay for online ads in one of two ways: pay-per-click advertising or fixed-rate advertising. Web content mining may provide insight into the behavior or intention of the user for using that Web page, and accordingly, relevant advertisements may be recommended.

Contents of a web page

Now, let us come to the implementation part. Web content mining deals with the content of Web pages, and the contents may be in an unstructured or semi-structured format. Web data comprises the majority of text data, which is unstructured. Hence, Web content mining usually relates to text mining. A Web document may have content in the following format:

- Text
- Images
- Audios
- Videos
- Lists
- Tables

Text, images, audios, and videos are unstructured data, whereas lists and tables are structured data. Depending upon the application, the content mining may be performed either only based on the text present in the Web document or based on images, audios, videos, or a combination of any of these contents.

As discussed earlier, Web content mining applies data mining for retrieving useful information from Web pages. Now, the problem is that Data Mining can only be applied when the data is structured.

Hence, the first step of Web content mining is transforming unstructured content into structured content. This act of transformation is called content pre-processing. Since the scope of this book covers text-based Web content mining, the content pre-processing or data pre-processing will be discussed with reference to textual data only.

Content pre-processing

Pre-processing of a Web page not only changes the format of contents but also converts erroneous data into proper data. Pre-processing involves various steps, but all the steps may or may not be applicable to a given task. During pre-processing, the Web content passes through three broad categories (Refer to *figure 2.3*).

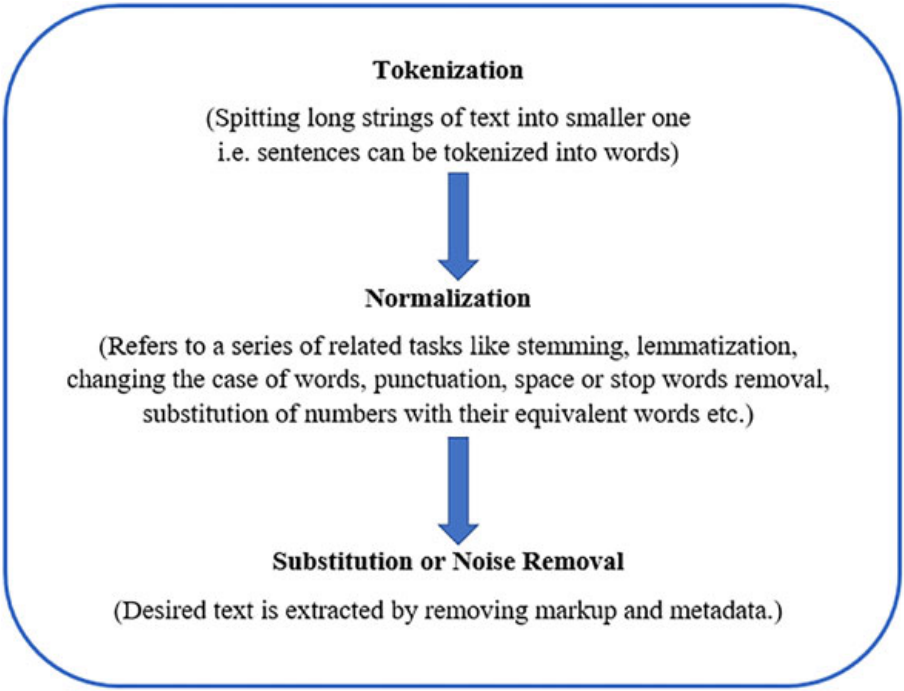


Figure 2.3: *The broad categories for pre-processing Web content*

The contents of the Web documents are wrapped in HTML or XML tags. So, the first step is always to extract the text by removing the markup tags. The second step is the cleaning of data which is done by filling in the missing values, removing the noise removing the redundant data. Missing values may be handled by any standard techniques, like using an average of the existing values or using the maximum/minimum appearing value in place of the missing value. The third step is tokenization. Tokenization is used to split long strings of text into smaller ones, which are called tokens. Tokens are useful or important words for that application domain. The next step is stemming. Stemming is performed by applying specific algorithms on the Web document, for example, the porter algorithm. Through stemming, similar meaning words are replaced by their root words; for example, if in a Web document, words like “laughing” and “laughed” appear, then these words can be replaced by the root word “laugh”. The next step deals with the removal of stop words, for example, words like a, an, the, such, as, to, in, and so on. Removal of stop words is important because these words have no meaning for that application domain. In the next step, **Collection Frequency** (**CF**) is calculated. The term collection frequency represents multiple occurrences of a significant term (t) in the collection. Now, per **Document Term Frequency** (**DFt**) is calculated.

DFt represents the number of times a particular term appears in the document. In the last step, the word document is represented by the words it contains, along with their occurrences. The steps of pre-processing are represented in *figure 2.4*.

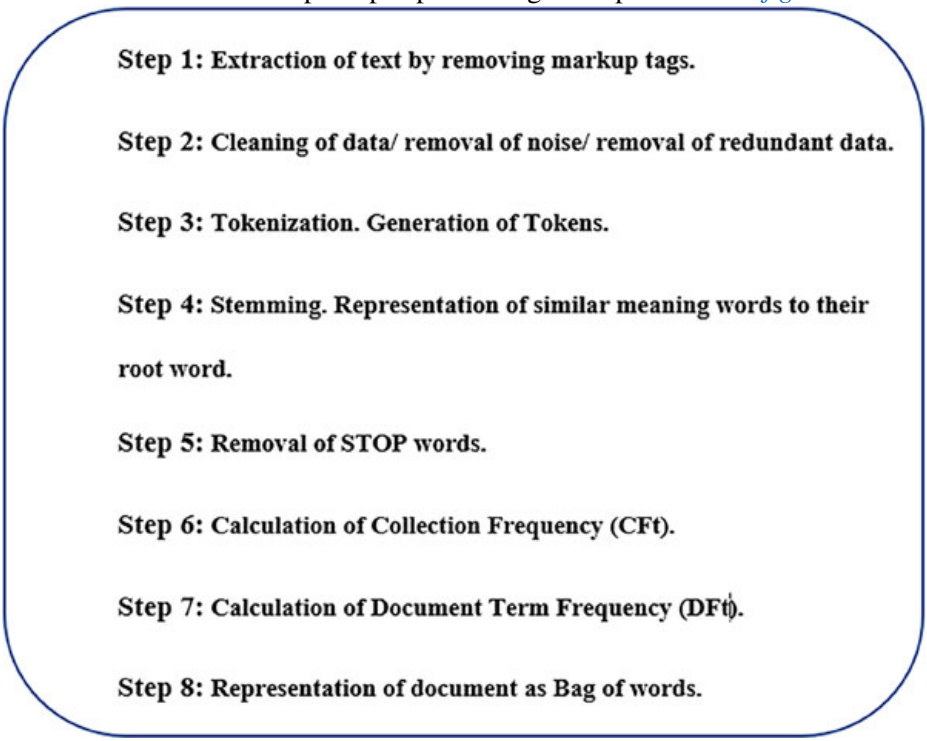


Figure 2.4: *Steps of Web document pre-processing*

Web content analysis

Once the Web document is pre-processed, the next step is to analyze the Web content as per the requirement of our application. Different data mining algorithms such as classification, clustering, association, and mining may be applied to the pre-processed structured data. *Table 2.1* illustrates Web Content Mining from the Information Retrieval view and Database view:

CHAPTER 3

Prominent Applications with Web Mining

Introduction

In today's business environment, Web-based applications are the best form of correspondence. Retail organizations have reclassified their business model and shifted to a Web-based model to improve the business yield. The business model over the Web gives a chance to clients and collaborators where their items and explicit business can be found. These days online business breaks the hindrance of time and space when contrasted with the actual office. Large organizations all throughout the planet understand that online business is not simply purchasing and selling over internet; rather, it improves the effectiveness to contend with different Goliaths on the lookout. For this reason, information mining, sometime called as information disclosure, is utilized. Web mining is an information mining strategy that is applied to the WWW. There are immense amounts of data accessible over the internet.

Structure

In this chapter, we will discuss the following topics:

- Personalized customer applications—E-commerce
- Web search
- Web wide tracking
- Personalized portal and Web
- Web service performance optimization
- Process mining
- Concepts of sequential pattern
- Concepts of association rules
- Association rule mining and Python libraries

Objectives

A business represented via websites provides online visibility and discoverability as well as establishes or enhances brand recognition. After studying this chapter, you should be able to understand the concepts of Web Personalization, Website Tracking, Optimization of Web service, Concept of Association Rules, and Sequential Patterns. These techniques play an important role when any business

shifts toward an online business model.

Personalized customer applications— E-commerce

Personalization is a dynamic search where part of the query comes from information about the user. We may understand personalized customer applications with an example like if you search for a hospital or police station on the net, then Personalized Customer Applications have become the demand of E-commerce. Companies understand very well that every visitor that interacts with them has their own specific needs, and for the benefit of their business, it is incredibly important that they meet the high expectations of their customers. Customers want a company that truly understands their needs, and they will not be afraid to look elsewhere if they feel that they are being treated with a one-size-fits-all marketing approach.

The demanding behavior of customers has given birth to different mechanisms through which the preferences of the user are collected via user's search behavior and Web-based activities over the World Wide Web. We will discuss Web search and Web tracking in the next section.

Web search

Web search denotes the search for information on the World Wide Web. In *Chapter 1, Web Mining—An Introduction*, while discussing about world wide Web, we discussed that several Web pages relate to other Web pages through something that is known as a hyperlink. We saw that the World Wide Web is a collection of billions of Web pages, and most of them are connected through hyperlinks to each other. These Web pages contain lots of relevant and related information. Let us say we want to search for some information about "Web Mining". There will be thousands of Web pages that contain information about this term. Now how to decide which page will give the most appropriate information? Web graph-related algorithms will help us to do so. To understand this, we need to have an understanding about Web graphs. The Web graph describes the World Wide Web as the directed graph between various pages on the Web. It consists of several nodes (the Web page) and directed edges (the links) connecting each Web page in the Web. The generated search results are generally presented in a line of results often referred to as **Search Engine Results Pages** (**SERP**).

The information may be a mix of Web pages, images, and other types of files. So, Web pages may contain information in the form of text, images, and videos. They also provide links to some other part on the same Web page or other Web page using their URL. These links are called hyperlinks and are the easiest way for navigation across the Web.

Figure 3.1 shows the linking of Web pages on the World Wide Web:

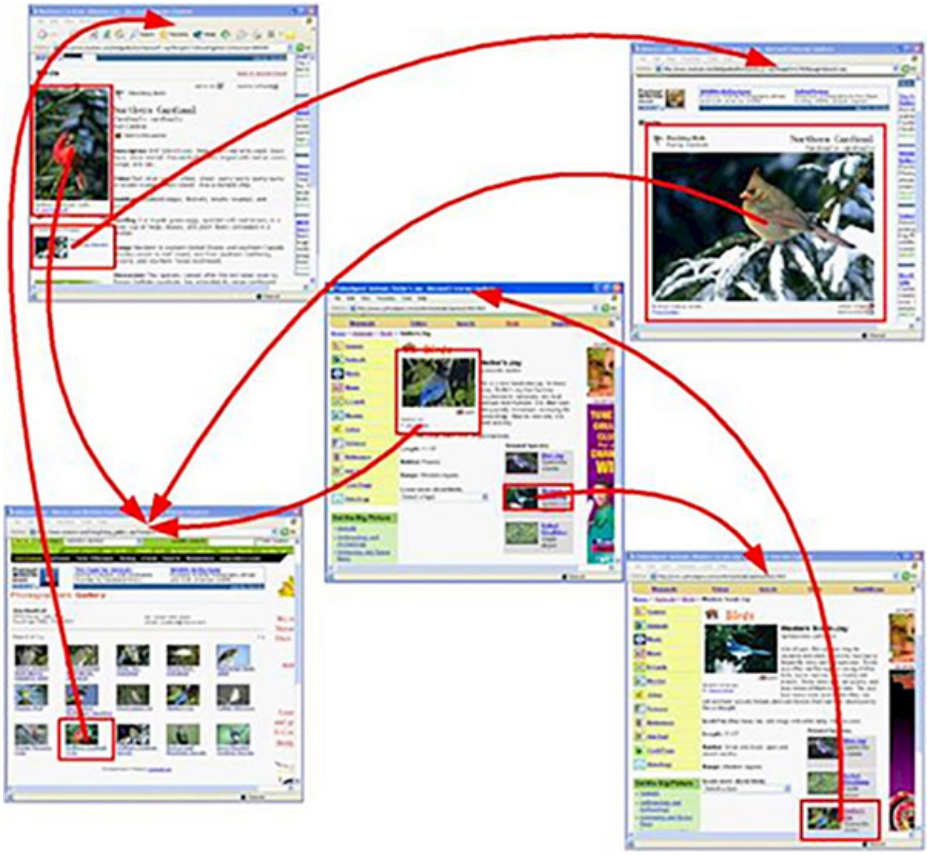


Figure 3.1: *Linking of Web pages*

Hyperlinks could be in the form of text or image. Moving your cursor over a hyperlink, its shape will become to pointing hand, as shown in *figure 3.2*:

Figure 3.2: *Hyperlink as text and image*

The Web has become a very resourceful entity in terms of information and is becoming a de facto database of the search for information. Searching information from the Web is often termed as Web search and may be understood as a search of information, images, video, multimedia content, and so on, on the Web using a search engine.

Web searches are performed using a software called Search engine. While searching the Web for a textual query, the Search engines perform the searching in an orderly manner. The output of a Web search is displayed to the user in the form of a list of pages that contain Web links to related Web pages, images, videos, and so on. Few search engines perform data mining from the database also.

Some search engines also use to crawl the pages containing the results using an

algorithm or crawler. Traditionally, information retrieval systems and search engines are used to extract the relevant documents based on the similarity of Web page contents, query entered, and indexed pages. In the 1990s, it was found that the methods based on content were not sufficient to get the best results as a large number of Web pages contained the given information, resulting in a large number of pages returned because of the query on search engines.

When you open a browser such as Edge, Firefox, or Chrome, they mostly present the search engine of their choice. For example, the default search for Microsoft edge is Microsoft Bing.

Many search engines being used in English-speaking countries are as follows:

- Ask Search engine
- Lycos
- Infoseek
- Google Search engine
- MSN Bing directory/search engine
- Yahoo! Directory/search engine
- Yippy Search engine, and so on
- Altavista

Google is so dominant that people tend to consider that this is the only search engine. For example, by giving the search string “Web Search” on the Google search engine, we get 13,76,00,00,000 results, as shown in *figure 3.3*. The results are based on the query, and the more specific the query better is the result:

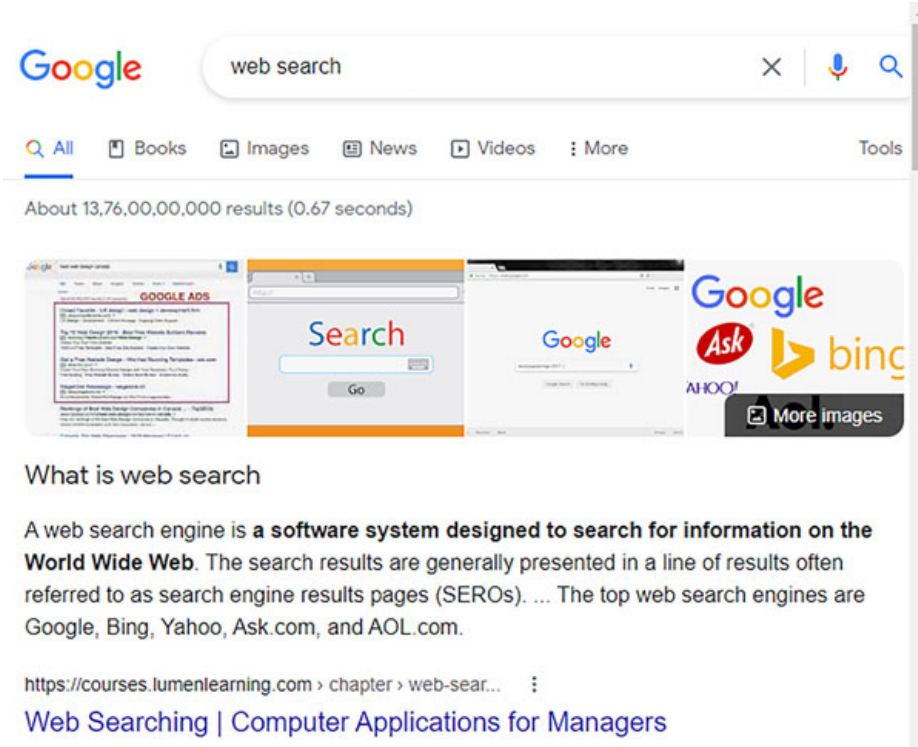


Figure 3.3: *Result of simple search*

Whereas if the query is "Web search in Web mining", it returns 55,90,00,000 pages, as shown in the following figure:

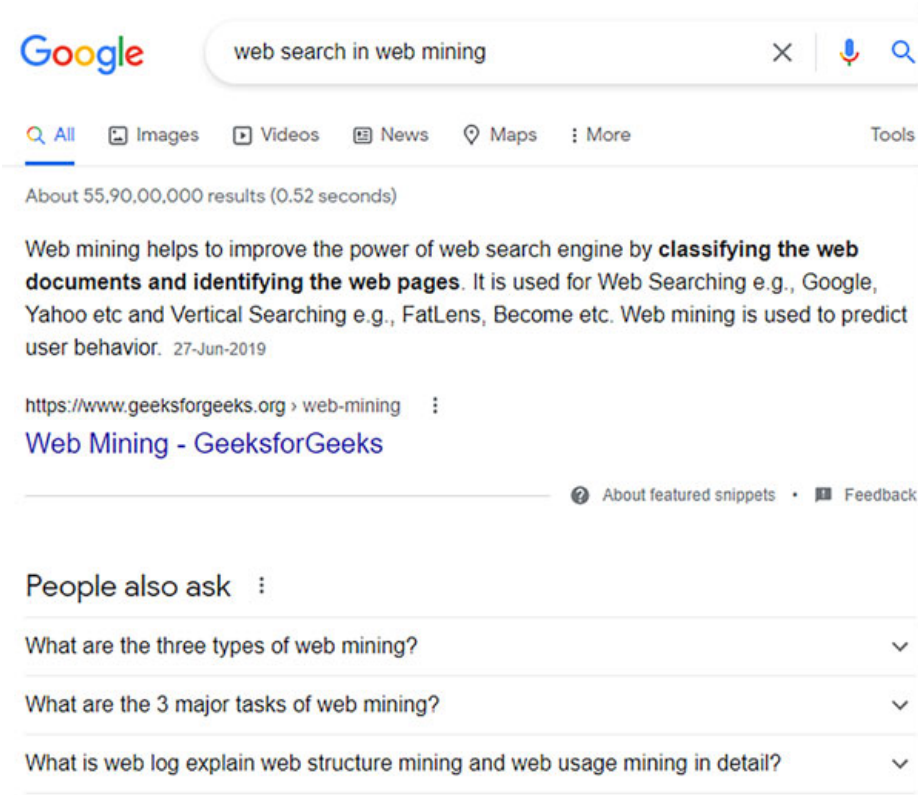


Figure 3.4: *Result of compound search*

To get the most appropriate pages, link analysis approaches that are based on hyperlinks are implemented. To compute importance of a Web page (node in a graph), hyperlinks-based algorithms PageRank and HITS were created in 1998. Both these algorithms, PageRank, and HITS, are used for ranking Web pages and are based on social network analysis, *Chapter 7, Web Structure Mining*.

These algorithms use hyperlink structure to give rank according to the degree of prestige or authority. One of the most popular search engines, Google, uses the PageRank algorithm which was developed by Google founders Sergey Brin and Larry Page in 1998. This algorithm provides a ranking to each Web page based on how many links are leading to that page from other sites (Inlinks) (for details, see *Chapter 2, Web Mining Taxonomy*. The basic idea behind the PageRank algorithm is that if a random Web surfer on visits a page and follows the links, they keep visiting linked pages (random walk). The popularity of any page is computed as the average number of times this random visitor has visited a particular page following the links. The probability that a page is visited by a random surfer on the Web is considered an important factor for ranking the search results. This probability is termed as page-rank, which is computed iteratively. In *figure 3.5,* Web pages are shown as a connected graph:

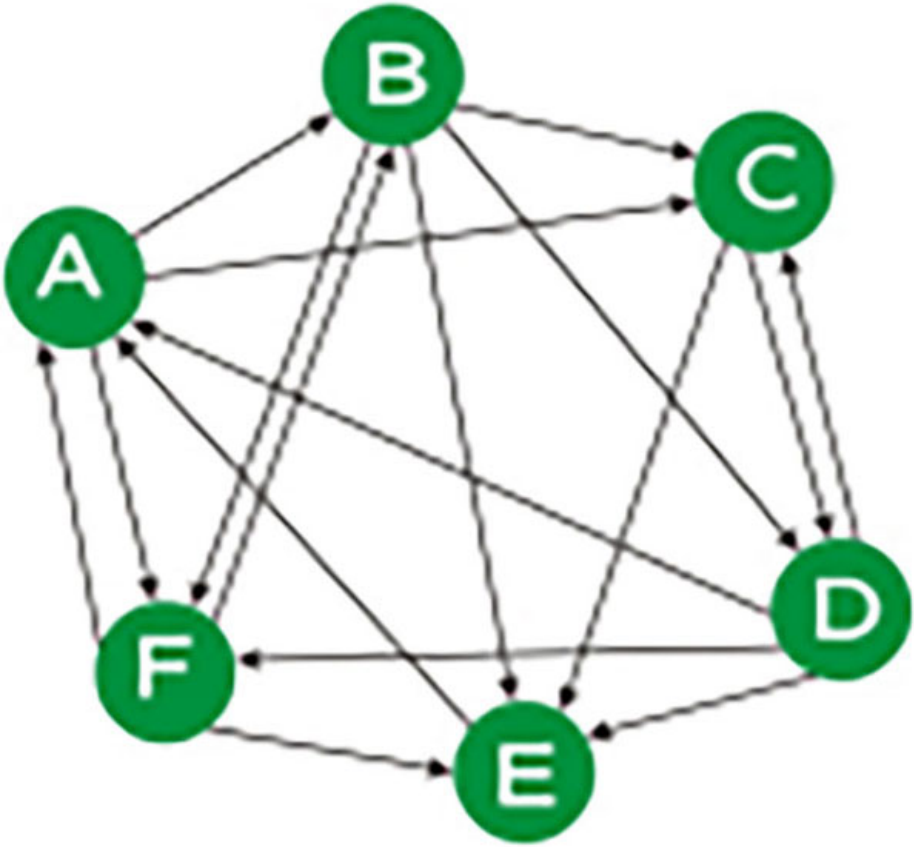


Figure 3.5: *Web pages as a connected graph*

Note that in *figure 3.5*, there are six Web pages that are connected with each other using links (represented as a directed graph). The number of times each page is visited through links from other pages (shown by Inlinks) is as follows:

```
Page A : 3
Page B : 2
Page C : 3
Page D : 2
Page E : 4
Page F : 3
```

Thus, the rank of page E, which is 4, is the highest, and thus, we can infer that Page E is the most visited page (PageRank algorithm has been discussed in *Chapter 2, Web Taxonomy*).

Web Wide Tracking Web tracking can be referred as part of visitor management. Basically, it is the practice through which visitor's activities over the World Wide Web are collected and stored by operators of websites and third parties collect. Nearly every company we can think of uses Web tracking technology to learn about and monetize our behavior and preferences.

Analysis of a user's behavior may be used to provide content that enables the operator to infer their preferences and may be of interest to various parties, such as advertisers.

With the help of Web tracking technologies like cookies, end user's actions are collected. The collected actions are processed, and accumulated statistics on the interest, behavior, preferences, and movement of the end-user are presented.

A website tracking tool will track, measure, and report on website activity and visitor behavior, including traffic, user clicks, and performance (that is, conversion rate). With the help of tracking tools, you can monitor what is happening on your website, and you can accordingly design your action plan to optimize for improved UX and business goals.

A few popular free Web tracking tools are as follows:

- **Google Analytics**: For measuring website traffic and finding the best/worst performing pages.
- **Hotjar**: For tracking user behavior and gathering product experience insights.
- **Google Optimize**: Uses A/B test page variations (two or more variants of a page are shown to users at random, and statistical analysis is used to determine which variation performs better for a given conversion goal) to find the best-performing designs.
- **Google Search Console**: Measures SEO performance and check for crawling and speed errors.
- **Mixpanel**: For monitoring product metrics like activation, retention, and churn.

Web tracking is classified into two categories: first-party and third-party. When the sites directly collect the information of visitors, then such type of Web tracking is referred as the first party of Web tracking. An example of such type of Web tracking is the collection of IP addresses or page visits.

When the user's activities through multiple sites are tracked by another site or server, then it is termed as third-party tracking. For example, if a user visits bestbuy.com and clicks on a product, third-party trackers will collect and analyze the information about that user and their activity on bestbuy.com.

Most common methods of website tracking

In this section, we will discuss four basic types of website tracking:

IP tracking

Any device connected to the internet can be uniquely identified by its IP address. Although the IP address is visible to everyone, the personal information of the user is not disclosed. But if a website can specifically determine where the device is

located; for example, according to the city or region the targeted campaigns can be launched.

Cookies

Cookies are files that are stored on the user's device (mobile or computer) by the website. The file contains information regarding the online activities performed by the user while surfing a website. Currently, more than 40% of websites use Cookies. Cookies permit sites to remember the preferences of the user and personalize their experience when the user revisits the site.

Fingerprinting

Websites use special scripts to collect enough information about us when we visit any website. JavaScript is a common language used for writing such code. These codes run in the background of the browser. These codes run in the background of the browser. They collect information about the user based on the basis of device specifications, and a unique profile for each visitor is created by the website. Device specifications such as operating system, size of the screen, IP address, language settings, and so on are collected.

Tracking pixels

In tracking pixels, the tracking code is not visible to the user as the code is embedded within the image.

When the website or image, which contains tracking pixels, is opened by the user, the embedded code is executed and starts tracking the user's activity. The tracking pixel, which is also called as Web beacon, is basically an image file, usually a transparent GIF.

Personalized portal and Web

Do you know the basic difference between a Website and a Portal? Let us have a quick review of it. A Web site is a collection of various Web pages and is accessed with a URL. URL specifies the unique location of a website. Web portals are also stored on the internet and are accessed via a specific URL. Websites provide access to contents to all the end users, whereas Web portals provide login-protected access to the user. Web portals do provide user-specific content.

Tatnall, 2005 mentioned that "*A Web portal is a gateway to the information and services on the Web where its users can interchange and share information*". The fundamental principle for enabling personalization in a Web portal is based on the client-side knowledge: the more knowledge a Web portal has about its user and the user's environment, the more sophisticated personalization features could be provided to the user. Though people who access a Web portal are all so similar in their interests then also a standardized way of providing information that fits to the

need of all users is not feasible. And that is why Personalization of Web Portals is required.

Different key factors play an important role while the technique for personalizing the Web portal is decided. Different factors such as device, operating systems, and user agent profiles are used for Improving searches on Web Portals. As the scale of information on the Web portals increases, so a systematic approach is required to develop, deploy, and maintain them. With the proper use of security, personalization can be put into practice using a variety of techniques, including the use of cookies, rules based on click stream monitoring, content caching, the use of recommender systems, collaborative filtering, and implicit/explicit user profiling.

Web service performance optimization

The speed of the website makes the first impression about your business. It is crucial to realize that if you were not able to impress the user for the first time and his experience was not positive, and you will not get a second chance. Low website speed is one of the most frustrating things that will indeed turn people away regarding your resource. Slow websites will cost you money and damage your reputation. Web applications are a mixture of server-side and client-side code. Your application can have performance problems on either side, and both need to be optimized. Web performance optimization follows monitoring and analyzing the performance of Web applications and identifying ways to improve it. Web performance optimization offers many benefits like an increased rate of conversation, ranking of search engines, and improved page views. The performance optimization also lowers the server bandwidth costs. To one of the most popular search engines—Google, has adopted a new methodology to protect its users from low-performance website. For this, even for desktop searches, the ranking is being done based on mobile versions of pages.

While analyzing the performance of a website, three main website metrics are considered. These are bounce rate, average time on the page, and unique visitors.

Bounce rate

Bounce rate is single-page sessions divided by all sessions or the percentage of all sessions on your site in which users viewed only a single page and triggered only a single request to the Analytics server. As a rule of thumb, a bounce rate in the range of 26%–40% is excellent. A total of 41%–55% is roughly average.

Average time on page

Average time in page is a Web analytics metric that measures the average amount of time spent on a single page by all users of a website. Average Time on Page is calculated as the total time on page, divided by the total number of pageviews minus the number of exits.

Unique visitors

A unique visitor is a term used in marketing analytics that refers to a person who has visited the website at least once and is counted only once in the reporting time period. Unique visitors are a key metric when building a website's following or selling advertising space because it shows how many people are visiting the site. So, if a user visits a site for more than one time, the count will not exceed; it will only exceed if a new user visits the site. Even if different users are accessing the site with the same IP address, the count will remain the same. The official Google Analytics definition of this term is: "Unique Visitors are the number of unduplicated (counted only once) visitors to your website over the course of a specified time period".

Websites with high performance will have lower bounce rates, increased conversions, and better end-user experiences resulting in a higher return rate of visitors.

The time taken by a page to show on the user's screen is represented by a metric referred to as Page load time. This is obvious that if the page load time is reduced, it will influence the promotion and sales processes.

In a research report by HubSpot, it has been mentioned that even a 1-second delay in page load may result from a 7% reduced conversion. It is interesting to note that if the Amazon page is slowed down for 1 second, it could cost $1.6 billion in sales per year.

We have discussed the metrics used for measuring the performance of a website (bounce rate, average time on page, and unique visitors) now, and we will discuss the key factors, which effect the success of the website. So, the factors effecting the success of a website are as follows:

- **Conversion**: means getting your visitors to do what you want them to do.
- **Visibility**: Website speed is one of the factors that Google takes into consideration when ranking sites. The load time of a website will impact how easily users can find that website.
- **Usability**: The better a website performs, the more satisfied the user will be.

As the Web applications are a mixture of server-side and client-side code. Your application can have performance problems on either side, and both need to be optimized. The client-side relates to performance as seen within the Web browser. The server side relates to how long it takes to run on the server to execute requests.

Let us discuss a few strategies for optimizing client-side performance.

- **Caching:** The static files of a website, such as JavaScript, CSS, and image files, should be cached on servers closer to where the users are. This will indeed increase the performance of the website. The traffic from the server will be offloaded, and the site will load faster.
- **Bundel**: The content of files can be optimized by using minification

techniques as well as by having fewer files; these files may be bundled together. One point should be taken care, and that is bundling, and minification typically requires code changes to complete.

- **Optimization of Images**: Images should be made smaller to achieve optimization. If the website is using small icon files, then instead of them, icon fonts may be used. Another good way could be the lazy loading of images. In this technique, while loading a page, only those images should be loaded that can be seen on the screen, and additional images should be loaded as the screen is scrolled.

Process mining

Process mining applies data science to find, approve, and further develop work processes. By joining data mining and process intelligence, organizations can mine log information from their data frameworks to figure out the presentation of their cycles, uncovering bottlenecks and different areas of progress. To remain competitive, online retailers need to adapt in an agile, non-structured way, resulting in large, unstructured websites and rapidly changing server resource demands

In process mining, a process is a chain of events made up of process steps with a clear start and end activity. Process Mining algorithms use data from so-called event logs. Those are not typical log files but different types of databases and files from enterprise information systems or enterprise resource planning systems. Process mining aims to extract a business process model from a set of execution logs. Process mining algorithms typically and all activities (nodes) in the process model, constructing a dependency graph with no edges, and then search through the space of process models by adding, deleting, and reversing edges.

Process mining basically bridges the gap between classical process model analysis and data-oriented analysis like data mining and machine learning. Process mining uses real-time data and focuses on processes. But at the same time, using real data.

There must be one question in your mind, and that is why do we need process mining? Or What is the purpose of doing process mining? So, the answer is that in classical data mining, one, on average, does not look at processes. Especially not end-to-end processes. And if one is concerned with process model analysis, then the data will be completely ignored. So, the answer to the question is that we are doing process mining to respond to performance-related questions and compliance-related questions.

Concepts of association rules

Associations are relationships between objects. For beginners, the simplest explanation of Association Rule is “Association rules are “if-then” statements that help to show the probability of relationships between data items within large data sets in various types of databases.

Association rules were introduced in 1993 by Agrawal et al. These rules are an important class of regularities in data. Mining of association rules is a fundamental data mining task. It is possibly the most significant model developed and comprehensively studied by the database and data mining community. Its objective is to find all co-occurrence relationships, called associations, among the data items.

Association rule mining can be understood as an unsupervised learning technique. The technique tries to discover some interesting association or relation among the given event, variable, or data set. Here, the mapping between the given items is done based on their dependencies. If the occurrence of an event or a data item depends on another event or data item, it will be mapped accordingly so that the rule becomes useful.

The typical application of association rule mining is the famous market basket data analysis, which aims to determine how one item purchased by customers is associated with another item. An example association rule is

Bread ® Butter [support = 10%, confidence = 80%].

The previously-mentioned rule says that 10% of customers buy Bread and Butter together, and those who buy Bread also buy Butter 80% of the time. Support and confidence are two measures of rule strength.

By this example, we may conclude that—an association rule consists of a set of items, the rule body, leading to another item, the rule head. The association rule relates the rule body with the rule head.

Association rule learning is a machine learning technique used for discovering interesting relationships between variables in large databases. For any given multi-item transaction, association rules aim to obtain rules that determine how or why certain items are linked.

There are various types of association rules as follows:

- **Multi-relational association rules**: each rule item consists of one entity but several relations.
- **Generalized association rules**: Provides the basic idea of interesting patterns hidden in data. Since the patterns are extracted at each level of abstraction, it results in larger rule sets. The use of this big-sized rule set for effective decision-making is a challenging task. Rules should have categorical (nominal or discrete) properties on both the left and right sides of the rule.
- **Quantitative association rules**: Quantitative association denotes association with item sets and their quantities. To find such association rules involving quantity, we partition each item into equal spaced bins, with each bin representing a quantity range. One attribute (left or right) of quantitative association rules must contain numeric attributes.
- **Interval information association rules**: The standard formulation of association rules are suitable for describing patterns found in each data set.

Several difficulties arise when the standard rules are used to infer about novel instances not included in the original data. Data is first pre-processed by data smoothing and mapping. Next, interval association rules are generated, which involve data partitioning via clustering before the rules are generated using an Apriori algorithm. Finally, these rules are used to identify data values that fall outside the expected intervals.

Association rule mining

The idea behind association rule mining is to determine rules that allow us to identify which objects may be related to a set of objects we already know. A common example of association rule mining is basket analysis, as shown in the following figure:

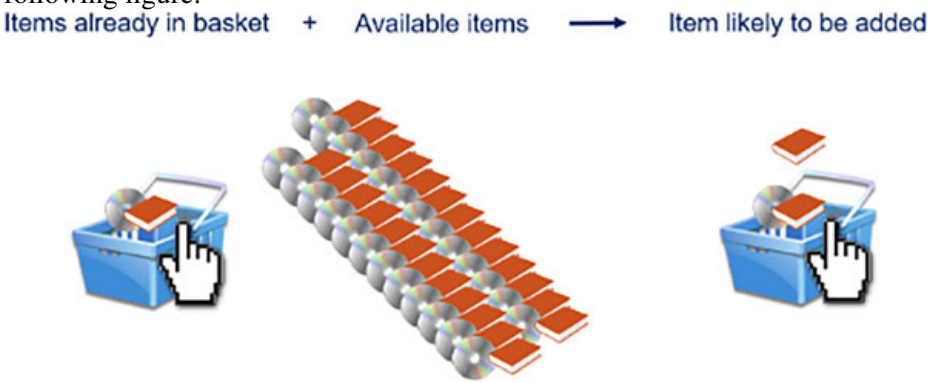


Figure 3.6: *Example basket analysis*

In Association rule mining, we have a database referred as *transactions database*, containing a list of transactions, and in each transaction, we observe a set of related objects. We apply association rule mining to a set of transactions to infer *association rules* that describe the associations between items:

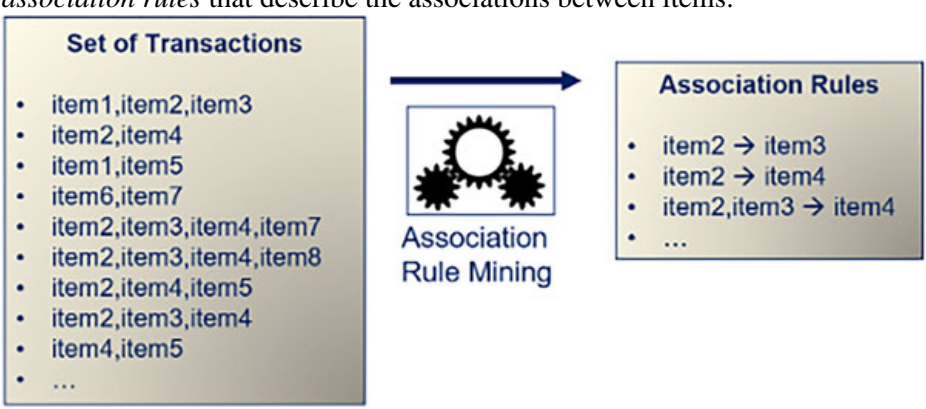


Figure 3.7: *Association rule mining*

The goal of association rule mining is to identify good rules based on a set of transactions. A generic way to define “interesting relationships” is that items occur

often together in transactions.

Let us discuss it through an example; suppose we have a database with 10 transactions.

```
[ P r o d u c t 1 P r o d u c t 2 P r o d u c t 4 ' ] ,
[ ' P r o d u 2 c t , ' P r o d u 4 c t ] ,
[ ' P r o d u 1 c t , ' P r o d u 5 c t ] ,
[ ' P r o d u 6 c t , ' P r o d u 7 c t ] ,
[ ' P r o d u c t 2 P r o d u 3 c t , P r o d u 4 c t , ' P r o d u 7 c t ] ,
[ ' P r o d u c t 2 P r o d u 3 c t , ' P r o d u 4 c t , ' P r o d u 8 c t ] ,
[ ' P r o d u 2 c t , ' P r o d u 4 c t , ' P r o d u 5 c t ] ,
[ ' P r o d u 2 c t , ' P r o d u 3 c t , ' P r o d u 4 c t ] ,
[ ' P r o d u 4 c t , ' P r o d u 5 c t ] ,
[ ' P r o d u 6 c t , ' P r o d u 7 c t ] ]
```

The given transaction shows that items Product2, Product 3, and Product4 occur often together. This shows that a useful relationship may exist between these items.

Now the next task is how to find such combinations of items automatically and to design good rules from the combinations of items.

The answer is Apriori Algorithm. Apriori Algorithm is the most used and very powerful algorithm used to perform Association Rule Mining.

Components of Apriori algorithm

There are the following three major components of Apriori algorithm:

- Support
- Confidence
- Lift

Support and frequent itemsets

The algorithm is based on the concept of the *support* of itemsets IS⊆I. The support of an itemset IS is defined as the ratio of transactions in which all items i∈IS occur, that is:

$$support(IS) = \frac{|\{t \in T : IS \subseteq t\}|}{|T|}$$

As previously discussed, finding the combinations of frequently existing items automatically is a much-required task. The Support directly measures how often a combination of items occurs. For building rules, we need to decide a minimum value for considering any combination of items interesting from any given itemset. Now, the itemsets which are having a value of Support greater than this threshold will be treated as frequent itemsets.

Formally, we call an itemset frequent IS⊂I, if `s u p p o r t ( I S ) ≥ m i n`

minimal required $support \in [0, 1] min\{0, 1\}$

Confidence

Confidence refers to the likelihood that Item B is also bought if item A is bought. It can be calculated by finding the number of transactions where A and B are bought together, divided by the total number of transactions where A is bought.

Confidence(A→B) = (Transactions containing both (A and B))/(Transactions containing A)

Lift

Lift(A -> B) refers to the increase in the ratio of sale of B when A is sold. Lift(A –> B) can be calculated by dividing Confidence(A -> B) divided by Support(B).

Lift(A→B) = (Confidence (A→B))/(Support (B))

Refer to the previously-taken transaction example, the item's Product 2, Product3, and Product4 would have support ({Product2, Product3, Product4})= 1/10

You may easily find out the reason, and it is because all three items occur together in three of the ten transactions. Thus, if the value of minsupp is taken as 0.3, {Product2, Product3, Product4} will fall in the category of frequent itemsets. Overall, we find the following frequent item sets, as shown in the following table:

Table 3.1: *Frequent itemset*

Steps in apriori algorithm

Following are the steps for Apriori Algorithm:

1. Since the Apriori algorithm intends to find the rule for the relation between the items based on the association between items. To confirm the association between items at the first step, a minimum value for support and confidence is set. Here, the support means that rules will be selected for those data items, which have certain existence. Similarly, the confidents represent the minimum required value for co-occurrence with other items.
2. Now all those subsets will be extracted which are satisfying the first condition, that is, the value of support is higher than the threshold value set by us at the beginning.
3. Once the extraction is done, all the rules having a higher value of confidence from the threshold value will be selected from the subset.
4. Finally, the rules will be ordered. Ordering of rules is done by arranging the rules in descending order of Lift.

Concepts of sequential pattern

We live in a world where businesses collect vast amounts of data daily. With so

much data available, there comes a need to analyze data and get insights to be able to make sound decisions.

We experience a daily reality such that organizations gather immense measures of information every day. With such a lot of information accessible, there comes a need to break down information and persuade bits of knowledge to have the option to use that information for better decisions.

Sequential pattern mining, a subset of data mining, is the method involved with recognizing frequently occurring subsequences as the pattern. Sequential pattern, so basically, sequential patterns are frequent patterns that are available in one or several successive transactions of many input sequences. Finding Sequential patterns is highly useful for Bioinformatics, retail, telecommunications, health care, fraud detection, and so on.

We perform Sequential pattern mining to find the pattern between data examples that are statistically relevant and where the values are presented in a sequence or group.

The basic difference between Association rule mining and sequential pattern mining is that in the former, one does not have the sequence of items set in consideration. Whereas in the sequential pattern, the order of the sequence of items set in any transaction is significant.

There are several areas in which sequential patterns can be used, such as Web access pattern analysis, weather prediction, production processes, and Web intrusion detection.

During Sequence Pattern Mining, the following tasks are performed effectively:

- Building efficient indexes for sequence information
- Frequently occurring patterns are discovered
- Sequences are compared
- Finding missing sequence items

Let us discuss some important concepts related to sequential pattern mining:

Sequence database

A sequence database consists of ordered elements or events. Although discussing Association Rule mining, we have discussed about Transaction Database. So, now the question comes into the picture is what is the difference between Transaction databases and Sequence database? *Figure 3.8* depicts the basic difference between both.

Transaction Database

TID	itemsets
10	a, b, d
20	a, c, d
30	a, d, e
40	b, e, f

Sequence Database

SID	sequences
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

Figure 3.8: *Difference between Transaction databases and Sequence database*

Here, we can see clearly that in the transaction Database, the itemset represents list of items, whereas, in the Sequence Database, the item sets are presented as an ordered collection of elements or events. Sequence Database may be represented as a set of tuples <SID, S>. Here, SID represents the Sequence ID, and S represents the Sequence.

Subsequence versus supersequence

The following points depict the difference between subsequence and supersequence:

- A sequence is an ordered list of events denoted < e1 e2 … el >.
- Given two sequences α=< a1 a2 … an > and β=<b1 b2 … bm >.
- α is called a subsequence of β, denoted as α⊆β, if there exist integers $\leq j <$ $j <\ldots<$ jn $\leq$ m such that a1 ⊆ bj1, a2 ⊆ bj2,…, an ⊆ bjn.
- if α subsequence of β, then β is a super sequence of α.

Let us discuss one example to clear the concept. Let us take a sequence: *A = <(abcd), (gh), (yz)>* and *B = <(abcd), (efgh), (lmn), (xyz)>*. Here, the point to be noted is that every element of A is a subset of at least one unique element of B and is in the correct order. Thus, A is the subsequence of B and B is the supersequence of A.

Minimum support

To perform sequential pattern mining, two things must be ensured. First is that a sequence database must be provided, and the parameter called the minimum support threshold must be specified.

This parameter minimum support threshold denotes when a pattern should be considered as frequent and should be shown to the user. Basically, it is a minimum number of sequences in which a pattern must appear.

Let us discuss one example, suppose you set the minimum support threshold to 3, so now all sub-sequences occurring in at least three sequences of the input database

will be extracted.

You can see that in the example database, 29 sub-sequences met this requirement. For a given sequence database:

SID	Sequence
1	$\langle\{a,b\},\{c\},\{f,g\},\{g\},\{e\}\rangle$
2	$\langle\{a,d\},\{c\},\{b\},\{a,b,e,f\}\rangle$
3	$\langle\{a\},\{b\},\{f,g\},\{e\}$
4	$\langle\{b\},\{f,g\}\rangle$

Figure 3.9: *Sequences satisfying the minimum threshold value*

The table given as follows shows the sequential patterns. The right column of the table indicates the support, that is, the number of sequences containing each pattern.

Pattern	Sup.
$\langle\{a\}\rangle$	3
$\langle\{a\},\{g\}\rangle$	2
$\langle\{a\},\{g\},\{e\}\rangle$	2
$\langle\{a\},\{f\}\rangle$	3
$\langle\{a\},\{f\},\{e\}\rangle$	2
$\langle\{a\},\{c\}\rangle$	2
$\langle\{a\},\{c\},\{f\}\rangle$	2
$\langle\{a\},\{c\},\{e\}\rangle$	2
$\langle\{a\},\{b\}\rangle$	2
$\langle\{a\},\{b\},\{f\}\rangle$	2
$\langle\{a\},\{b\},\{e\}\rangle$	2
$\langle\{a\},\{e\}\rangle$	3
$\langle\{a,b\}\rangle$	2
$\langle\{b\}\rangle$	4
$\langle\{b\},\{g\}\rangle$	3
$\langle\{b\},\{g\},\{e\}\rangle$	2
$\langle\{b\},\{f\}\rangle$	4
$\langle\{b\},\{f,g\}\rangle$	3
$\langle\{b\},\{f\},\{e\}\rangle$	2
$\langle\{b\},\{e\}\rangle$	3
$\langle\{c\}\rangle$	2
$\langle\{c\},\{f\}\rangle$	2
$\langle\{c\},\{e\}\rangle$	2
$\langle\{e\}\rangle$	3
$\langle\{f\}\rangle$	4
$\langle\{f,g\}\rangle$	3
$\langle\{f\},\{e\}\rangle$	2
$\langle\{g\}\rangle$	3
$\langle\{g\},\{e\}\rangle$	2

Figure 3.10: *Sequence pattern and support*

If the threshold value is set to 2, we can see that 29 subsequences met this requirement. The patterns are frequent and have a support of 3 and 2 sequences, respectively. Sequential pattern mining can be applied to a database containing hundreds of thousands of sequences.

Prefix and suffix

If all elements of a sequence are in alphabetical order, then a sequence *S' =<s1',s2',..,sm'>* is called a prefix of sequence *S = <s1,s2,…sn> if si' = si* for *i <= m-1 and sm'* is a subset of sm.

For example, given a sequence <a(abc)(ad)d(ef)>, its prefixes are <a>, <aa>, <a(ab)>, <a, (abc)> and so on. The part of the sequence after the prefix is called the suffix or postfix to the prefix.

Projection

If A and B are two sequences such that B is a subsequence of A, then a subsequence A' of A is called a projection of A w.r.t. B if:

- A' has the prefix B.
- There exists no proper supersequence A'' of A' (that is, A'' not equal to A' and A' is a subsequence of A'') such that A'' also has the prefix B.

Thus, the projection of <a(abc)(ac)d(ef)> w.r.t. <ab> is <(_c)(ac)d(ef)>. The projection of <a(abc)(ac)d(ef)> w.r.t. <ad> is <(ef)>.

Association rule mining and python libraries

For performing association rule mining in Python, we will need two Python libraries: **pandas** and **mlxtend**

Pandas

Pandas is an open-source Python package that is most widely used for data science/ data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays.

Mlxtend

Mlxtend (**Machine Learning Extensions**) is a Python library of useful tools for day-to-day data science tasks. You can install the Mlxtend package through the Python Package Index (PyPi) by running pip install Mlxtend.

If you do not have them installed, please open "Command Prompt" (on Windows) and install them using the following code:

```
pip install pandas
pip install mlxtend
```

1. Create a dataset with the required data.

```
dataset = [
    ["Milk", "Eggs", "Bread", "Butter"],
    ["Milk", "Butter", "Eggs", "Apple"],
    ["Milk", "Eggs", "Bread"],
    ["Bread", "Butter"],
    ["Eggs", "Apple"],
]
Print(dataset)
```

Following is the output screen:

```
[['Milk', 'Eggs', 'Bread', 'Butter'], ['Milk', 'Butter', 'Eggs', 'Apple'], ['Milk', 'Eggs', 'Bread'], ['Bread', 'Butter'], ['Eggs', 'Apple']]
```

Figure 3.11: *Result*

2. Convert list to dataframe with Boolean values.

```
import pandas as pd
from mlxtend.preprocessing import
TransactionEncoder
te= TransactionEncoder()
te_arr=te.fit(dataset).transform(
df= pd.DataFrame(te_array, columns=te.colum
```

3. We first import the required libraries. Then we save the **TransactionEncoder** function as a local variable **te**. The next step is to create an array (**te_ar**) from the dataset list with True/False values (depending on if an item appears/does not appear on a particular receipt).

   We then convert this array to a **dataframe** (**df**) using items as column names.

```
print(df)
```

4. Will generate the following output:

	Apple	Bread	Butter	Eggs	Milk
0	False	True	True	True	True
1	True	False	True	True	True
2	False	True	False	True	True
3	False	True	True	False	False
4	True	False	False	True	False

Figure 3.12: *Output*

5. Find frequently occurring itemsets using Apriori Algorithm

```
from mlxtend.frequent_patterns import
frequent_itemsets
frequent_itemsets(df,
min_support=0.6,use_colnames=True)
```

6. First, we import the Apriori algorithm function from the library. Then, we apply the algorithm to our data to extract the itemsets that have a minimum

support value of 0.2 (this parameter can be changed).

7. To get the list of frequent item sets:

```
print(frequent_itemsets_ap)
```

	support	itemsets
0	0.4	(Apple)
1	0.6	(Bread)
2	0.6	(Butter)
3	0.8	(Eggs)
4	0.6	(Milk)
5	0.2	(Butter, Apple)
6	0.4	(Eggs, Apple)
7	0.2	(Milk, Apple)
8	0.4	(Butter, Bread)
9	0.4	(Eggs, Bread)
10	0.4	(Milk, Bread)
11	0.4	(Butter, Eggs)
12	0.4	(Butter, Milk)
13	0.6	(Milk, Eggs)
14	0.2	(Butter, Eggs, Apple)
15	0.2	(Butter, Milk, Apple)
16	0.2	(Eggs, Milk, Apple)
17	0.2	(Butter, Eggs, Bread)
18	0.2	(Butter, Milk, Bread)
19	0.4	(Eggs, Milk, Bread)
20	0.4	(Butter, Milk, Eggs)
21	0.2	(Butter, Eggs, Milk, Apple)
22	0.2	(Butter, Eggs, Milk, Bread)

Figure 3.13: *Resulted output*

8. Mine the Association Rules

In this final step, we will perform the association rule mining in Python for the frequent item sets, which we calculated in Step 3:

```
from mlxtend.frequent_patterns import association_rules
rules_ap = association_rules(frequent_itemsets_ap, metric="confidence", threshold=0.8)
```

	consequent support	support	confidence	lift	leverage	conviction
0	0.8	0.4	1.0	1.250000	0.08	inf
1	0.8	0.6	1.0	1.250000	0.12	inf
2	0.8	0.2	1.0	1.250000	0.04	inf
3	0.6	0.2	1.0	1.666667	0.08	inf
4	0.6	0.2	1.0	1.666667	0.08	inf
5	0.8	0.2	1.0	1.250000	0.04	inf
6	0.6	0.4	1.0	1.666667	0.16	inf
7	0.8	0.4	1.0	1.250000	0.08	inf
8	0.8	0.4	1.0	1.250000	0.08	inf
9	0.6	0.4	1.0	1.666667	0.16	inf
10	0.6	0.2	1.0	1.666667	0.08	inf
11	0.8	0.2	1.0	1.250000	0.04	inf
12	0.6	0.2	1.0	1.666667	0.08	inf
13	0.6	0.2	1.0	1.666667	0.08	inf
14	0.4	0.2	1.0	2.500000	0.12	inf
15	0.6	0.2	1.0	1.666667	0.08	inf
16	0.8	0.2	1.0	1.250000	0.04	inf

Figure 3.14: *Result of applying association rule mining*

First, we import the required function from the page to determine the association rules for a given dataset using some set of parameters.

Then, we apply it to the two frequent item datasets which we created in Step 3.

The `"metric"` and `"min_threshold"` parameters are set as per the requirement of the business problem.

Conclusion

This chapter provides a basic introduction to Web Search and various techniques involved to improve Web Search. We have discussed the concept of Web tracking, and we have understood how websites use Web tracking for visitor management. While discussing popular tools and common methods of website tracking, we saw that websites store device and user-specific information to provide personalized and customized services to users.

We also discussed in detail that the performance of any website can be measured by various metrics and Average time and Conclusion rate are the common ones.

We understood the many forms of Sequence Pattern Mining, the terminologies related to Sequence Pattern Mining, and the popular Apriori-based algorithms. Association rule learning is an important technique of machine learning, and it is used in Market Basket analysis, Web usage mining, continuous production, and so on. In market basket analysis, it is adequately used by several big retailers to find the relations among items.

Points to remember

- Web Search denotes the search for information on the World Wide Web. The information may be a mix of web pages, images, and other types of files.
- Web searches are performed using a software called Search engine.
- With the help of web tracking technologies like cookies end user's actions

are collected. A website tracking tool will track, measure, and report on website activity and visitor behaviour including traffic, user clicks, and performance.

- Google Analytics, Hotjar, Google Optimize, Google Search Console and Mixpanel are few free tools to manage web tracking.
- IP Tracking, Cookies, Fingerprinting and Tracking pixels are four basic types of website tracking.
- The speed of the website makes the first impression about the business. Web performance optimization follows monitoring and analysing the performance of web application and identifying ways to improve it While analysing the performance of a website, three main website metrics are considered. These are bounce rate, average time on page and unique visitors.
- Association Rule mining tries to discover some interesting association or relation among the given event, variable, or dataset.
- Sequential Pattern Mining, a subset of data Mining, is the method involved with recognizing frequently occurring subsequences as pattern.

Multiple Choice Questions

1. The analysis performed to uncover the interesting statistical correlation between associated-attributes value pairs is known as the _____.

    a. Mining of association
    b. Mining of correlation
    c. Mining of clusters
    d. All of the above

2. Which of the following is the correct depiction of Digital Marketing?

    a. E-mail Marketing
    b. Social Media Marketing
    c. Web Marketing
    d. All of the above

3. In what ways can site traffic help in assessing the market value?

    a. Overall site traffic can be followed, and a general idea of marketing's impact can be determined.
    b. There is no association between the site traffic and marketing.
    c. Ads can send receivers to a specific landing page, which can be tracked.
    d. Product sales from the company website can be attributed directly to the marketing campaign.

4. The hyperlink refers to a _____.

a. Inbound link
b. Outbound link
c. IFTTT link
d. KPI link

5. What is the correct abbreviation of SERP?

a. Search Engine Result Page
b. System Engine Random Page
c. Search Estimate Result Page
d. System Estimate Random Page

6. To identify the users, do Web analytics tools need to report on?

a. user sessions
b. Unique users
c. Page Views
d. All of the above

7. Which of the following method is used to identify user sessions?

a. IP address
b. authenticated user
c. user agent
d. All of the above

8. Which of these extracts information from user-generated content for further scrutiny?

a. JavaScript Tagging
b. A/B Testing
c. Web-scraping software
d. MROCs

9. Which ONE of the following is mainly used in Web Analytics and is free of charge?

a. Google Analytics
b. Radian6
c. AlteranSM2
d. Social Radar

10. To make your website mobile friendly, you can make your website

a. Responsive

b. Reactive

c. Fast Loading

d. Light

Answer

Questions

1. Explain the optimization of Web search.
2. How Web tracking is performed?
3. Explain applications of Association Rules

Key terms

- Web Search Optimization
- Website Tracking
- Personalized Web Portals
- Association Rule
- Web Process Mining

CHAPTER 4

Python Fundamentals

Introduction

Python is easy to learn as well as open-source programming language also, which makes it an ideal language for beginners. Python provides an advantage in that it helps in writing in fewer lines. It has a large set of standard libraries, such as NumPy, Matplotlib, Pandas, sci-kit learn, and so on, which makes the data analysis job quite easy. Python is thus becoming one of the favorable coding languages for data scientists. This chapter introduces the reader to the Python language, basic libraries used for Web mining, and installation of Python. In this chapter, we have demonstrated examples based on the default editor, IDLE, which is part of Python installation for beginners. But, in upcoming chapters in the book, many examples will be demonstrated using Google Colab or Anaconda **Integrated Development Environment** (**IDE**). Installation of Anaconda will be discussed in this chapter. This chapter also includes a discussion on HTML tags that may be of help in scraping. Let us begin with our journey of learning Python.

Structure

Topics to be covered in this book are as follows:

- Introduction to Python
- Basics of Python
- Basic HTML tags
- Basics of Python libraries
- Installation of Python
- Introduction to commonly used IDE's and PDE
- Installation of Anaconda

Objectives

After studying this chapter, you will be able to understand the basics of Python, HTML tags, commonly used Python libraries, installation of Python under Windows, UNIX and Mac, Common Python IDEs, and Installation of Anaconda. After reading this chapter, you will be able to write, save, and run Python programs easily. Also, you will be familiar with important Python libraries and various commonly used IDE's that are used for Python programming. This chapter also discusses the process of installing Python under Windows, LINUX, and Mac

environment.

Introduction to Python

There has been a recent drift toward the use of the Python programming language, a language developed by Guido van Rossum. Python is one language that is very simple yet very powerful. The main reason for its simplicity is easy syntaxes which give developers the flexibility to focus on problem-solving rather than the syntax, complexity, and structure of the language. Due to its ease of learning and usage, Python codes can be easily written. Its simplicity, efficient data structures, effective object-oriented programming, and powerful features make it suitable for making the language for scripting and rapid application development in most of the areas on all platforms. Its rich libraries and ability to develop applications in almost all domains make it more powerful.

Basics of Python

Python has become a very popular language because of its features. It has become a language of choice for both beginners and professionals. Some of the important features are listed as follows:

- **Simple:** Python is an easy-to-read and minimalistic language. Its syntax resembles a lot to the English language, making it easy to learn.
- **Easy to Learn:** Python syntaxes are very simple; thus, the language is very easy to learn.
- **Free and Open Source:** Python is based on the concept of **Free/Libre and Open-source Software** (**FLOSS**), which means that Python software and code can be freely distributed and allows to make changes to it. It is based on a community that shares knowledge and is thus constantly being improved by the community, making it suitable for all varied applications.
- **High-level language**: Python allows you to write code without bothering about the low-level details such as memory management and so on. by your program. All this is implicitly managed.
- **Portable:** Python scripts can easily be ported to almost all the available platforms without any change in them.
- **Interpreted:** Python is an interpreter-based language and can execute the program directly from the source code.
- **Object-Oriented:** Python supports both procedure-oriented and object-oriented programming. Python has a powerful but simplistic way of doing object-oriented programming, as compared to other object-oriented programming languages like C++ and JAVA.
- **Embeddable:** Python programs can be embedded within your C/C++ programs to enhance the capabilities of your program.

- **Extensive libraries:** Python's rich set of libraries help you to do various things such as unit testing, database, Web browser, machine learning, data science, scraping, and so on.

Python programming

This section introduces you to the basic concepts of Python programming. These concepts will help you to have a good foundation to help you understand the entire code used in this book. I am using **Integrated Development and Learning Environment** (**IDLE**), which acts as a basic IDE provided by Python, to explain the examples given in this chapter. Those who do not have IDLE or any Python interpreter installed can refer to the upcoming sections for installation of IDLE (Python interpreter) as well as Anaconda (a Python distribution that includes packages used for data science and Web scraping). We will discuss in detail about Anaconda in the section that discusses the installation of an Anaconda.

Python also provides all three constructs, sequence, selection, and iterations, like other programming languages. Also, modularity and reusability functions (both library and user-defined) are available in Python. Other data structures such as strings, lists, tuple, dictionaries, file handling, database management, and networking make Python an apt language for use. In this section, we will see a brief introduction to all the important concepts of Python that will help you to do Web scraping.

When you learn any language, the traditional "Hello World" program is the first program that we execute. There are two ways of using Python to run a program—using the interactive interpreter prompt or using a source file. In this section, we will see how to create a Python source file and run it.

If you are using Windows, then I suggest that you use IDLE, which is my preferred IDE for the beginner due to its ease of use. Though the code can be written in any text editor like notepad, it is a bad choice because it does not do syntax highlighting, and importantly, it does not support indentation of the text, which is very important in the case of Python scripts. Good editors will automatically help you do this. There are many good IDE's, and any one of them could be chosen. We are introducing Python programming on the Windows platform using IDLE.

For Windows users, you can use the IDLE program as follows:

1. Click on Start -> Programs -> Python 3.8 -> IDLE (Python GUI).
2. Click on File->New File (or press *Ctrl+N*) to open the editor to write a Python script. (Opens in a new window).
3. After writing the program, Click File->Save and name the file as `<filename.py>`. `.py` is the extension given to Python scripts.
4. To execute the code, Click run->Run Module or press *F5*.
5. The script will be interpreted and executed. If there is no error, it will be executed; else, the interpreter will tell you the line with the error. In such a

case, make appropriate changes in the source file and repeat Step 4.

Writing "Hello World", the first Python script

Traditionally, whenever you learn any programming language, the first program, you write and execute is "Hello World".

Open the editor and write the following code:

```
print("Hello World")
```

Save the source file and execute the code. You will get the following output:

Output:

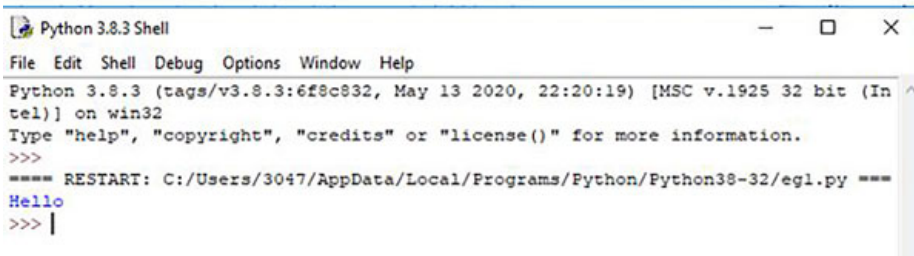


Figure 4.1: *Output screen of IDLE*

To exit the IDLE, press *Ctrl-z* followed by *Enter* or write `exit()` at the IDLE prompt. Make sure that you save and close the source file before you close IDLE.

Now, let us write a Python script with sequential statements having some input/ output statements and comments.

Program-1

```
1.# File name:mse1.py
2.i=int(input("Enter number "))      #input and convert to int
3.print(i)                  #the value of i
4.i=i+1
5.print(i)                  #incremented value of i
6.'''This is a multi-line string #giving multiline comment by enclosing
7.  This is the second line.'''
8.s="Welcome Python" #assign value to a variable
9.print(s)
```

Output:

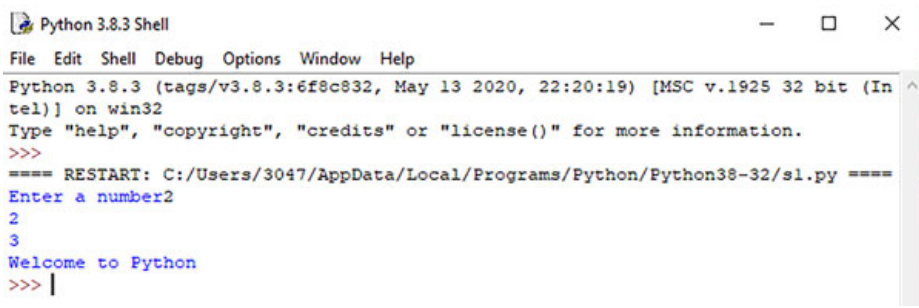


Figure 4.2: *Output of Program-1 showing input and output*

This program begins with a single-line comment, which is specified by "#". Python interpreter ignores and does not interpret anything that is written after "#" in Python. In the next line, we are taking input and storing it to the variable *i*. As the input entered is treated as a character, it should be converted to int using **`int()`** function in case any computation is to be done on such variables.

Conditional/selection statements

In sequential programming, the flow of statements is sequential. When it is desired to change in the flow of execution of statements, that is, in situations where a particular set of statements is to be executed based on some condition, it is important to make use of the selection statements available in Python. For example, we want to display "Good Morning" or "Good Evening" based on the time. I will introduce you to various selection construct available in Python that can be used to make decisions in the programs.

Python provides three control flow statements—if, for, and while. "If" is used for decision-making, whereas "for" and "while" is used for iterations.

The if statement

Conditions in Python are checked using the "if" statement. A block of statements is executed (called the if-block); if the condition is true, else we execute another block of statements (called the else-block). It is important to note that the else clause is optional, and we may have a statement only that the true condition is to be checked. A block of statements is recognized by the indentation of the statements.

For example:

Program-2

```
1 . i = 5
2 . j = 2
3 . i f ( i < j ) :                                  c # o n d i t i o n
4 .     p r i n t (i"s i s m a l l e r t h a n j " )     # b l o c k c u t i e f d
c o n d i t i o n t r u e
5 . e l s e :
```

```
6.      print(i,"is greater than",j)      #else block executed if
condition is false
```

Output:

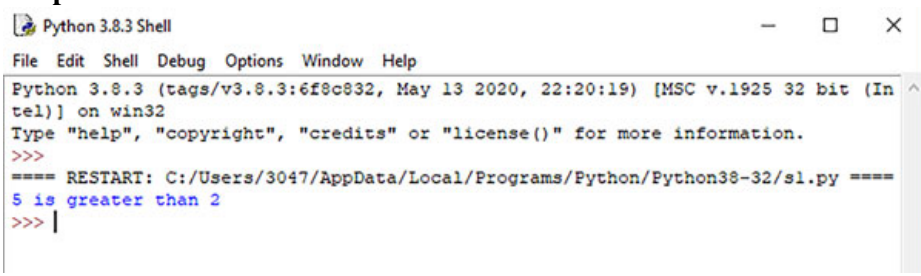


Figure 4.3: *Output of Program-2*

Note that the first line and the else line end with a colon (:) and the block of line that should be executed is indented forward. It is important to note that else is optional.

Multiple if conditions can be checked by if…elif ladder. Let us see one example of if…elif ladder.

Question: write a Python script that assigns a number to a variable and asks the user to guess it. Give appropriate hints if the number entered is greater than or less than the given number.

Program-3

```
1.number=32
2.choice=int(input('Enter integer:'))
3.if choice==number:
4.      print("Congratulations, you guessed it.")  If# block begins
5.      print('you won any prizes!')  If    # block ends here
6.elif choice<number:
7.      print('Not a little high than that')
Else block
8.      # You can have few more statements in this block
9.else:
10.      print('Not a little low than that')
11.      # you must have guess>number to reach here
12.print('Done')
13.
```

Output:

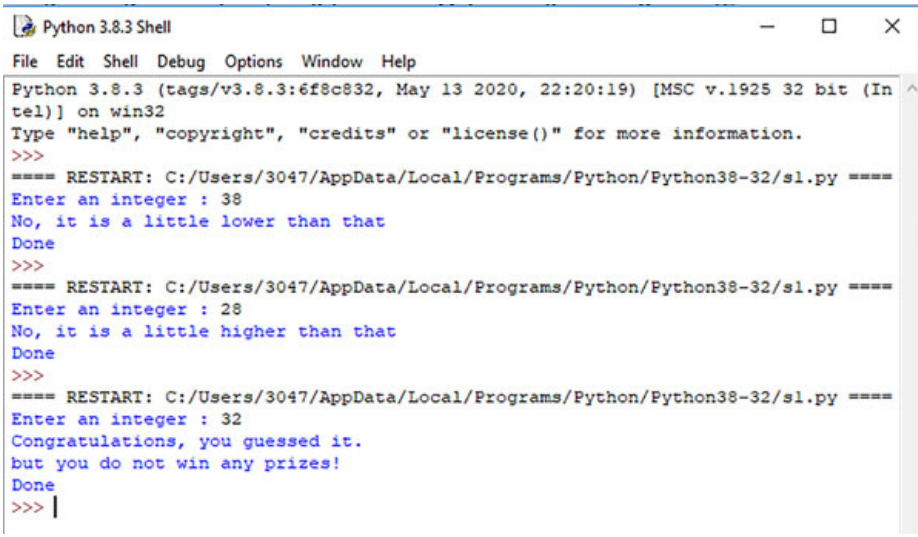


Figure 4.4: *Output of Program-3*

In this program, we initialize the value to a variable and ask the user to guess a number, and we then see if the user guesses the correct number. If the user guess is equal to the given number, print ("Congratulations, you guessed it."), else we check if the guessed number is less than the given number, print "No, it is a little higher than that". If none of the preceding conditions is met, the third condition is checked if the guessed number is greater than the given number, print "No, it is a little lower than that".

Point to remember

Using if…elif ladder can check multiple conditions.

- After each if, else or elif a colon needs to be given.
- Each block needs to be indented forward.
- We use "Else" clause to specify what is to be done if none of the conditions are true.
- Else if optional

Note: For C/C++ Programmers

It is important to note at this that there is no switch statement in Python. However, if…elif…else statement can be used to do the same thing.

Looping/iterative constructs

Whenever there is a need to repeatedly execute a block of statements, Looping/iterative statements are used to do so. Python provides the while and for statements for iterations.

While loop

"While" statement is the most used looping statement that allows executing the block of statements if a condition is true. The loop terminates as soon as the condition becomes false. C or C++ programmers may note that in Python, "while" statement may use an optional else clause.

Example:

Program-4

```
number = 32
running = True

while running:
    choice = int(input('Enter an integer : '))
    if choice == number:
        print("Congratulations, you guessed it.")
If block begins
        print('but you do not win any prizes!')
If block ends
        running = False # while loop will stop as
running becomes 'False'
    elif choice < number:
        print('No, it is a little higher than
that') End block begins
    else:
        print('No, it is a little lower than that')
else:
    print('The while loop is over.')
print('Done')

```

Output:

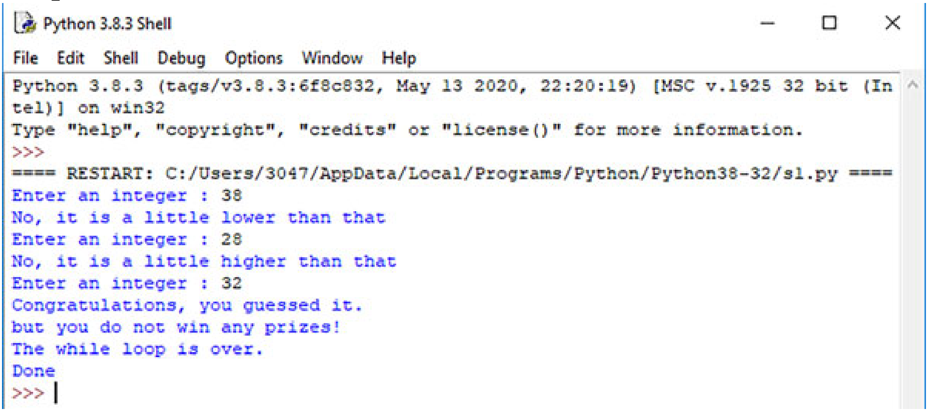


Figure 4.5: Output of Program-4

Remember, in the "while" loop also, the "else" clause is optional, and thus, can be used only if it is needed.

For Loop

Another looping statement, for…in the statement, iterates over a sequence of objects; that is, the loop will execute for each item in a sequence (the sequence is an ordered collection of items).

Example:

Program-5

```
number = 32

for i in [1, 2, 3, 4, 5]:
    guess = int(input('Enter an integer : '))
    if guess == number:
        print("Congratulations, you guessed it.") # New block begins
        print('but you do not win any prizes!') # New block ends
        break
    elif guess < number:
        print('No, it is a little higher than that') # Another block
    else:
        print('No, it is a little lower than that')
else:
    print('The for loop is over.')
print('Done')

```

Output:

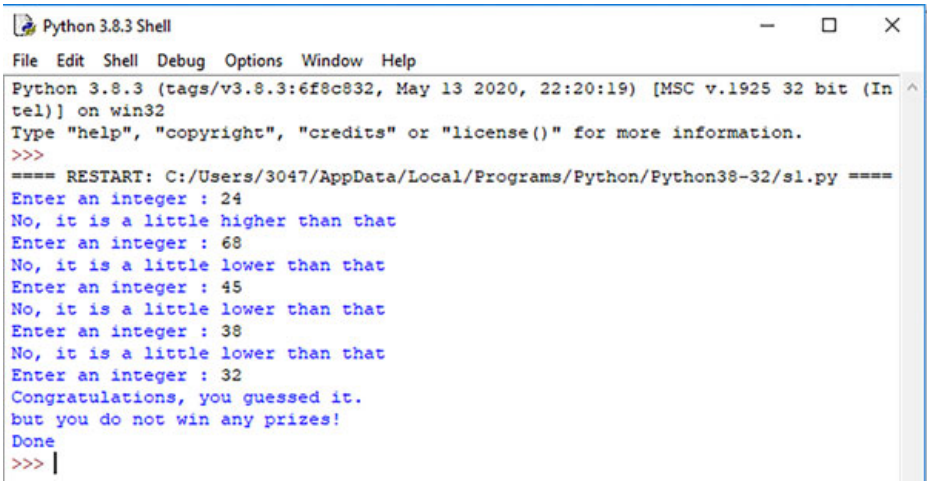


Figure 4.6: *Output of Program-5*

In this example, we see that for loop executes the block of statements for each value listed in the list [1,2,3,4,5]; that is, it will have five iterations. Note that by using break, we are ensuring that the controls move out of the loop as soon as the guessed number matches with the given number.

Note: It is important to note that the else part is optional. If present, it is always executed once after the for loop is over unless a break statement is encountered.

For loop also uses **range()** function to specify the elements, if they are in order, to specify the number of iterations.

Example:

Program-6

```
for x in range(1,5):
    print(x)
print('The loop is over')
```

Output:

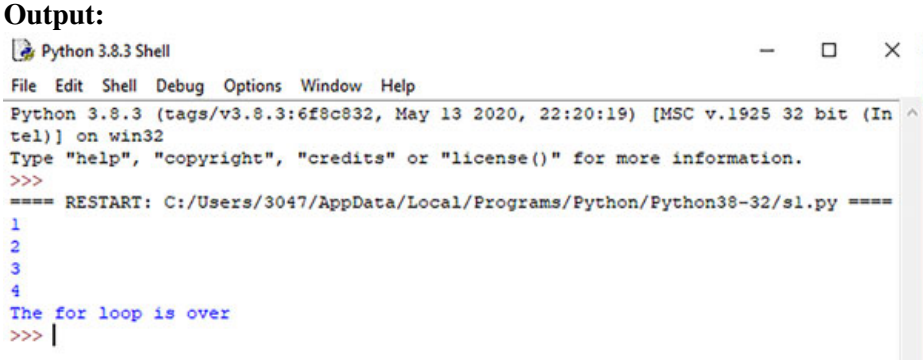


Figure 4.7: Output of Program-6

While using range() function, the last element in range is exclusive thus 5 is not printed.

Note: range() has other variants also. Those interested to learn about it may read Python documentation.

Functions

A very important concept, function, is a block of code or a piece of program that is written to make the program reusable and modular. Such blocks of codes are identified by a name that is known as a function name. Functions are categorized as built-in functions or libraries and user-defined functions.

Built-in functions are the functions that are built-in and are provided as library functions in any programming language.

User-defined functions are the functions that are defined by a user.

Functions are one of the most important building blocks in any programming language. We will now learn how to create and use user-defined functions in Python. We will also explore a few of the library functions and their usage.

Functions are defined using the def keyword. After this, the keyword comes as an identifier name for the function, followed by a pair of parentheses, which may enclose some names of variables, and by the final colon that ends the line. Next follows the block of statements that are part of this function. Let us take one example to understand how to define/ create a function.

Program-7

```
1.def print_hello():
2.    # function begins here
3.    print('Hello World')
4.    # End of function definition
5.
6.print_hello() # Function call
7.print_hello() # Call it again
```

Output:

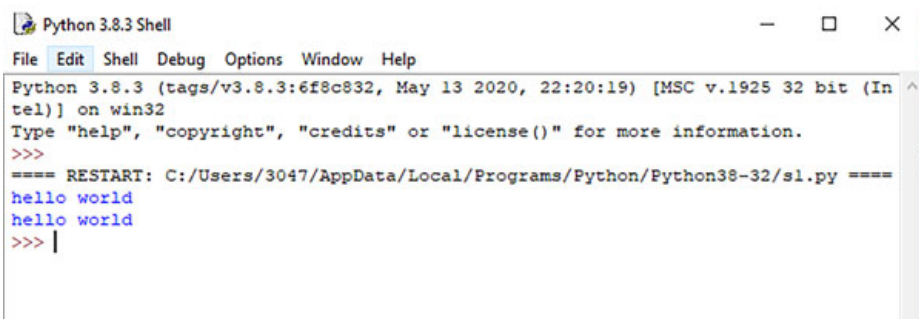


Figure 4.8: *Output of Program-7*

Note: The function body is defined once and has been called twice. A function can be called as many times as needed in a program.

Function parameters

In the previous example, we saw that every time we call the function, the same output is displayed. To make the Python program more flexible and dynamically behave based on input received from the main Python program, Python allows us to pass input variables to the function that are known as parameters.

The function uses these variables as input and then, after processing, returns the result. Parameters are passed within the bracket and are separated by a comma during the function call. At the receiving end also, the parameters are received within the brackets and separated by a comma.

Note: The names given in the function definition are called parameters, whereas the values you supply in the function call are called arguments.

For example:

Program-8

```
1 .d e fg r e a t e r ( a , b ) :
2 .   i f ( a > b ) :
3 .     p r i n t ('ai,sg r e a t te hr a n ' , b )
4 .   e l s e :
5 .     p r i n t ( b ,g 'r ie sa t te hr a n ' , a )
6 .
7 .x = 5
8 .y = 7
9 .g r e a t e r ( x , y )          c a# F w i n t c h t r i g o u n m e x n a t n d
y ( v a r i a b l e s )
1 0 .g r e a t e r ( 3 , 4 )          c a# F w i n t c h t c i r o s n t a n t
a r g u m e i t a s n d 4
1 1 .
```

Output:

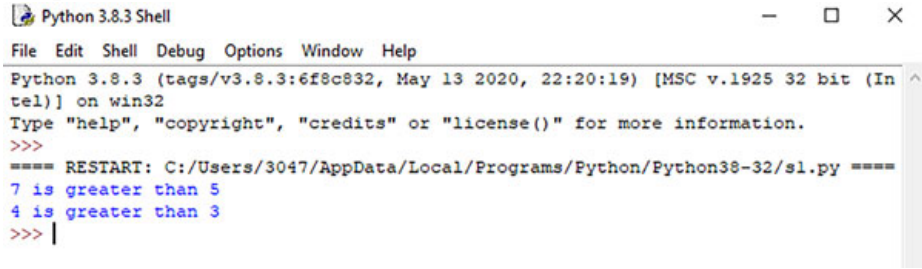


Figure 4.9: *Output of Program-8*

Notice that the parameter can be passed as a literal or a variable. Through the function call, the value of *x* and *y* or 3 and 4 is passed in each function call, and the function receives the values in arguments *a* and *b*. If block, then finds the greater value and print it in the function itself. It is also possible to return the value from the function back to the main program using the return keyword.

For Example:

Program-9

```
1 .d e fg r e a t e r ( a , b ) :
2 .   i f ( a > b ) :
3 .     r e t u an
4 .   e l s e :
5 .     r e t u bn
6 .
7 .x = 5
8 .y = 7
9 .m a x = g r e a t e r (#xF,uyn)c t ci e a h W i t h a r g u m exn t
a n dy ( v a r i a b l ree s t) u, r nveadl u s e t o r e i d h m a x
1 0 .p r i n t ( m aixsg r e a t e r ' )
1 1 .p r i n t ( g r e a t e r i(s3g,r4e)a,t'e r #F u n c t i e a h l
w i t h o n s t aanrtg u m e n ta s n d4 , v a l u r e e t u r n i e n d
p r i nstt a t e m e n t
1 2 .
```

Output:

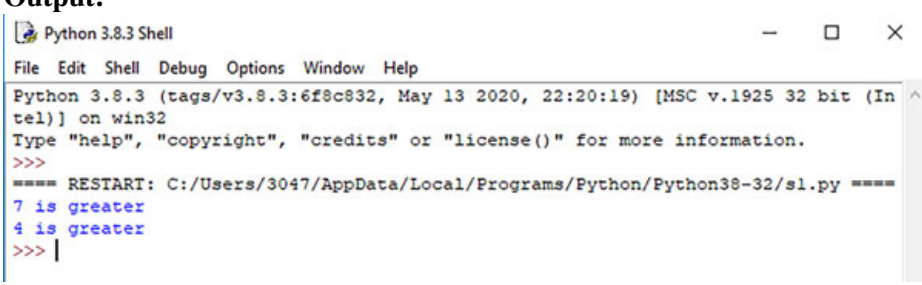


Figure 4.10: *Output of Program-9*

Notice that the value is returned using the "return" keyword and used in the "print" statement in the "main" program. In this example, we are using it to print the output, but it can be used for computation purposes also.

Note: Two different ways by which returned value has been received. The returned value can be stored in a variable or can directly be used in the expression.

Also, it is important to mention that any number of parameters that can be passed to the function depends on the number of variables defined in the function definition (the number must be the same).

Lists

The list is another very important data structure available in Python. A list may be considered as a collection of elements or sometimes known as items. An interesting thing about the list is that unlike an array (in other programming languages), a list can have a collection of data elements of the same or different data types.

A list can be defined by enclosing elements or items in a square bracket, that is []. Let us see some examples:

```
Fruits=['Apple','Mango','Banana']
```

Here, the list name is "Fruits", which has three elements "Apple", "Mango", and "Banana", each is of string types.

```
even=[40,22,24,98]
```

Here, the list name is "even", which has four integer elements 40, 22, 24, and 98.

```
emp=['Ram','Sales',30,50000]
```

This is an example of the list name "emp", which contains the employee name (Ram), department (Sales), age (30), and salary (50,000). This list contains elements from different data types.

An interesting aspect is that a list can have another list as its element. For example, we have a list that has elements as follows:

```
L=[[2,4,5],10,'XYZ',50000,85.9]
```

Here, the first two elements are of a list type.

List elements can be accessed by their position, known as index position. You write the list name followed by the square bracket and index position inside the square bracket. The first element is at index 0, the second at index 1, and so on. So, the last element is at index n-1. Let us try to access and print the list and its elements.

```
>>>print(emp)
['Ram','Sales',30,50000]
>>>print(emp[0])
```

```
Ram
>>>print(L[0])
[2,4]
>>>print(L[0][1])
4
```

The list in Python is mutable; that is, its elements can be modified in place, and the order of the elements can be changed. You just need to specify the index position of the element that needs to be changed.

```
>>>num=[1,3,5]
>>>num[0]=1
>>>print(num)
[1,3,5]
```

Note:

- If you try to access an index that does not exist, you will get an IndexError error.
- If an index has a negative value, then it counts backward from the end of the list.

"in" operator can be used to traverse through the list using the for a loop. For example:

```
for i in fruits:
  print(fruits[i])
```

Will print all the elements of the list.

Though there is a lot more to discuss regarding the list, that is out of the scope of this book. In this section, we have discussed all important aspects related to the list that may be of use in Web Scraping. If any other concepts are needed, they will be explained as and when it is required in the examples.

Basics of HTML: inspecting a Web page

The first step to scraping is to select the website you want to scrape. After opening the website, to inspect it, right-click anywhere on the page and choose "Inspect Element"/"View Page Source". You will notice that it shows a lot of information using **HyperText Markup Language** (**HTML**) tags.

Since we are going to perform scraping on the Web documents, it is important to understand the basic structure of a Web page. HTML is the commonly used language for the development of Web pages. Some of the common tags that are used to create a Web page are listed here in *table 4.1*:

CHAPTER 5
Web Scraping

Introduction

Web scraping is a technique by which unstructured data can be "collected" from the internet. Unstructured data can be understood as data that does not follow a template of existing data models of database concepts and could be found in varying formats. Unstructured data could be in the form of text, images, audio, video, or any combination of these that are stored as per the requirement of the website. Data available on the internet does not follow any specific format and is available in different formats on different websites. Web scraping is a very helpful technique that helps in collecting/extracting a large amount of data for analysis or research purposes from the internet, which is a huge repository of such data. Data scientists or engineers can scrap and store the data in a database or spreadsheet for later retrieval or analysis. Extracting such information can help businesses to take effective decisions. Some Python modules help us to do this job easily. This chapter begins with the basics like what is Web scraping, its uses, how a Web scrapper works, and its components, followed by topics like Python modules used for scraping, legal aspects related to scraping, data extraction and preprocessing, handling of text, images, and videos, scraping dynamic websites and CAPTCHA. In the end, the chapter demonstrates a case study implementation using Python.

Structure

In this chapter, we will discuss the following topics:

- Web scraping
- Use of Web scraping
- Working of Web scraper
- Web scraper and its components
- Python modules used for scraping
- Legality of Web scraping
- Data extraction and preprocessing
- Handling text, image, and videos
- Scraping dynamic websites
- Dealing with CAPTCHA
- Case study

Objectives

In this chapter, we will learn about **Web scraping**, its uses, components and how a Web scraper works. The chapter also discusses Python modules used for Web scraping, the legality of Web scraping, data extraction, and data preprocessing. It then explains the extraction of different types of data, that is, text data, image data, and video data. In the end, the chapter discusses about dealing with dynamic websites, CAPTCHA and dealing with CAPTCHA, followed by a case study implementing the learnings.

Introduction to Web scraping

Web scraping, often referred to as Web harvesting or Web extraction, is a technique that is used to extract useful information from the **World Wide Web** (**WWW**). The information extracted is saved to a file or a database for later use in its analysis. Web scraping techniques being used are diverse in nature, and it can be done manually or using automated tools like a Web crawler or a bot. Web scraping techniques being used today range from smaller (ad hoc) ones created for a specific application or fully automated systems that convert complete unstructured websites to a structured database.

Web scraping

World Wide Web is the largest repository of data, most of which is unstructured. In case any information is needed from a website, it can anytime be copied and pasted to the file for any kind of analysis. But it will contain all the data displayed on the Web page and will thus have a lot of data that is of no use to us. For example, you want product information on all the products on your favorite website. Each e-commerce website has many categories and sub-categories; you will need to copy-paste the contents from Web page of each of the categories and sub-categories to have data on all the products. Also, it will have a lot of data that may be relevant or irrelevant to us. This makes it difficult for us to analyze the data. This approach is very inefficient for larger projects.

Some websites offer **Application Programming Interfaces (APIs)** to interact with websites and fetch some relevant data, but this approach to is not good enough due to two reasons.

- Not only do API's allow us to extract of only limited data but also API's may not be available for all the data that we want.
- Also, using API may require some programming skills. If APIs are not available for any task or are not able to provide all the information that we desire, Web scraping can be useful for such a task. Web scraping helps in all such situations and helps us to scrap or extract only the data that is relevant to us.

We can give a formal definition of Web scraping as follows:

"*Web scraping is a technological solution to collect data from website in a quick, efficient and automated manner*".

Or, we may say Web scraping helps to extract relevant data from websites and transforms and stores it in a structured format like a CSV file or database, which can then be used to analyze for making effective decisions.

The following figure depicts the Web Scraping Data flow:

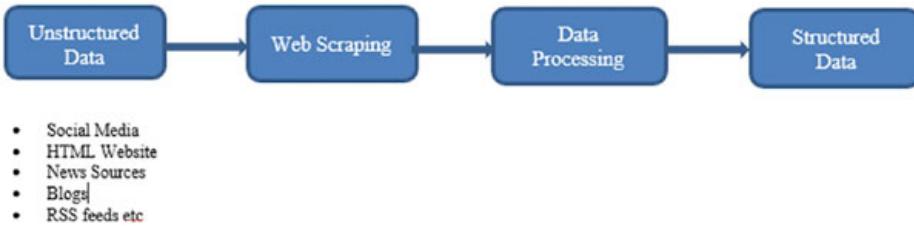


Figure 5.1: *Web scraping data flow*

Though it seems that Web scraping is a new concept, its historical roots in the time when **World Wide Web** (**WWW**) was born, the first Web robot was created in June 1993, to measure the size of the Web. From December 1993, search engines such as Infoseek, Altavista, Bing, and Google started the use of Web crawlers and Web bots. In the year 2000, companies like Salesforce and eBay launched their APIs with which the programmers could access and download some of the data available to the public. As seen earlier, API offers developers a friendly way to do Web scraping by just accessing the data provided by websites. But it has the limitation that one can fetch only the data provided by the website for the public; also, not all the websites provide API. So, programmers kept on working on a friendly approach that could facilitate Web scraping.

In 2004, the Beautiful Soup library was released, which was designed to be used in Python. Beautiful Soup library understands the site structure and helps in parsing HTML contents in a simple manner. It handles all the nitty gritty internally, providing a friendly method of parsing. It is considered to be the most advanced and sophisticated library for Web scraping. At the end of this chapter, we will see a case study of Web scraping using Beautiful Soup. In 2005–06, visual Web scraping was introduced with the intention that even non-programmers could do Web scraping on their own.

Uses of Web scraping

Web scraping or Web harvesting gives access to valuable data and has a lot of applications. Many businesses use Web scraping to improve their pricing, service, or operations. It provides a convenient mechanism for comparing information across various websites. Scraping websites helps in converting meaningful but unstructured data into a structured form where it can be used for various purposes. Say, for example, your query is to "To find the best-rated tablet in the lowest cost";

this can easily be done using a Web scraper (instead of scrolling through a lot of websites for comparison). Scraping can also be used for many other applications; a few examples of scraping are as follows:

- **Understanding trends**: Trend Analysis is a traditional practice of detecting patterns in any business for decision-making. Traditionally, trend analysis used to be done with company's own data, such as sales history or using publicly available datasets (such as the annual report of other companies or stock prices). However, due to the growing size of the availability of online data, businesses have the opportunity to collect business data in real time for better decision-making. Web scraping helps in automating the extraction of data for trend analysis, which saves a lot of time for data collection.
- **Change detection on a website**: Many websites change their content on a regular basis; for example, the price of a flight ticket keeps on changing regularly. It is not a good idea to regularly visit the website to check for changes. Web scraping program automates this process and can tell if the price is changing through a simple notification. Notification can also be customized to intimate only if the price has been reduced from the previous search. Its frequency can be decided to scrape every 15 min, an hour or daily as per the need.
- **Collecting data for research**: Researchers often find it challenging to obtain data for research purposes. Traditionally, various data collection methods were observation, surveys, interviewing, and so on. Nowadays, the internet has become a significant source of data for many professionals, scientists, and researchers. Instead of copying data from multiple sites and creating a dataset, Web scraping is very helpful in automating the task of data collection and creating datasets.
- **Competitor price monitoring**: In this era of E-commerce, monitoring competitors' prices is an important thing and plays an important role in product pricing. Manually keeping track of prices becomes almost an impossible task as the prices keep on changing regularly. Web scraping can be used to automate such tasks and helps in making appropriate strategies with respect to the product price and offering the products at the lowest price.
- **Aggregating news articles**: Web scraping is used to extract valuable input from different news stories, which can then be converted to actionable insights.
- **Real-time analytics**: Real-time analytics refers *to the process of preparing and measuring data as soon as it enters the database, so effectively* real-time data analytics can be understood as an analysis of data in real-time, that is, analysis of data as soon as it is available. Real-time analytics allows us to react without any delay and can be of great use in cases where an automated alert is raised when a particular event occurs.
- **Predictive analytics**: Data has a lot of hidden patterns, and extracting data

over a period of time can be used to understand these hidden patterns. Analysis of such data can be used to predict future outcomes or trends.

- **Lead generation**: Generating business leads is one crucial task in any business, and companies spend a lot of money to get these leads. Web scraping provides a cost-effective solution to provide quality leads.
- **Reputation monitoring**: Knowing what customers think about any product is vital for any company. In an online platform, it becomes difficult to see ratings and product reviews for all the products. Web scraping can be used to automate feedback collection and analysis.
- **Job portal**: Job portals have lots and lots of job-related posts. Web scraping can be used by job portals to scrape job-related information from various websites.

Working of Web scraper

Websites are designed to be read by humans, not machines. Web scraping may look like a simple process, but it actually requires complex procedures internally due to the complex and dynamic nature of the Web. But in general, it will follow the following process:

1. The Web scraper will be given one more URL (from where the data is to be retrieved).
2. The scraper program sends the GET request to the server and returns a response in the form of Web content.

   For example, to retrieve data from URL **https://www.myntra.com/girls-dresses**, this URL is provided to the scraper, which sends a GET request to the Myntra.com Web server, which, in turn, returns the HTML code of the following Web page as shown in *figure 5.2:*

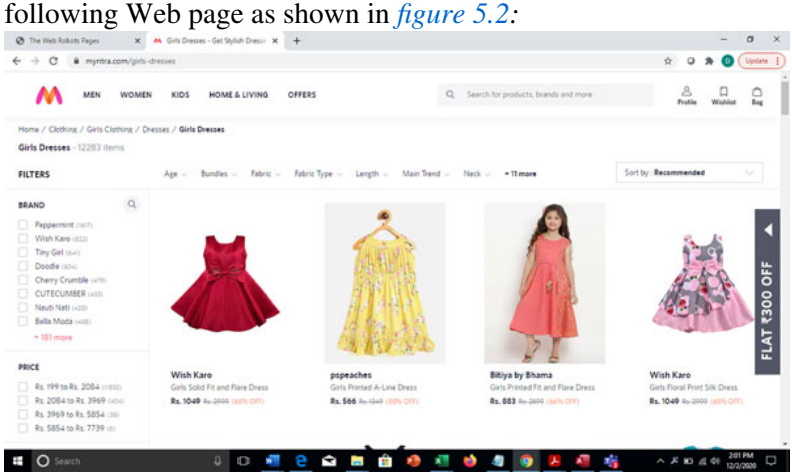


Figure 5.2: *Accessing Web page using URL*

3. More advanced scrapers return the entire website, including CSS and

JavaScript elements. This code is parsed following the tree structure path.

4. The server returns the data in HTML format. Now, the scraper retrieves either all the data from the Web page or only the desired data as specified by the user (For example, from Myntra product page, you may be interested only in the Product Model and its Price, not the reviews). This is accomplished using a process known as parsing. "Parsing" is breaking down the block of data (string and so on) into pieces or segments and converting the desired data into the format the script needs.
5. Extracted data is then stored in some structured format from where the user can easily do the analysis. This format could be in CSV format or an Excel sheet, or any other desired format.

The data of the website is depicted as follows:

Figure 5.3: *Data of website (unstructured) to an excel database (structured)*

A Web scraper may use different techniques for Web scraping as discussed as follows:

- **Traditional copy and paste**: This is the traditional method that was earlier used to scrape desired data from websites. This method suffers two major drawbacks: one, it fetches all the data (though most of the part is not relevant), and one must look for the relevant data from the fetched data; two, it needs human intervention and may take a lot of time when the large volume of data is to be scrapped. Also, it introduces errors as it is tiresome to analyze and store lots of data.
- **Hypertext Transfer Protocol (HTTP) Programming**: Using this technique, the scraper may extract data from static and dynamic pages. Data can be retrieved using HTTP requests to extract data from remote servers using socket programming.
- **Hyper Text Markup Language (HTML) Parsing**: In this technique, page content can be retrieved using semi-structured data query languages like XQuery and Hyper Text Query Language (HTQL) to parse HTML pages.
- **Document Object Model (DOM) Parsing**: In this technique, scraping is done by embedding a full-fledged Web browser, such as Internet Explorer or Mozilla browser, controls to retrieve the dynamic content generated by client-side scripts. These Web browser controls parse Web pages into a DOM tree, based on which programs can retrieve parts of the pages.

- **Web scraping software**: This technique uses the existing Web scraping tools, which can be customized as per the requirement. These tools are designed in a manner that they automatically recognize the structure of the page and generate the code for scraping, eliminating the need for coding. They also provide functions to write data into local databases. The only drawback to such tools is that they may not give us the exact data that we may desire.
- **Computer vision Web page analyzers**: This technique is used to identify Web page structure through the visual analysis of Web pages as images. This work is still under research and is expected to use computer vision for identifying Web structure.

The scraping process can be shown as depicted in the following diagram:

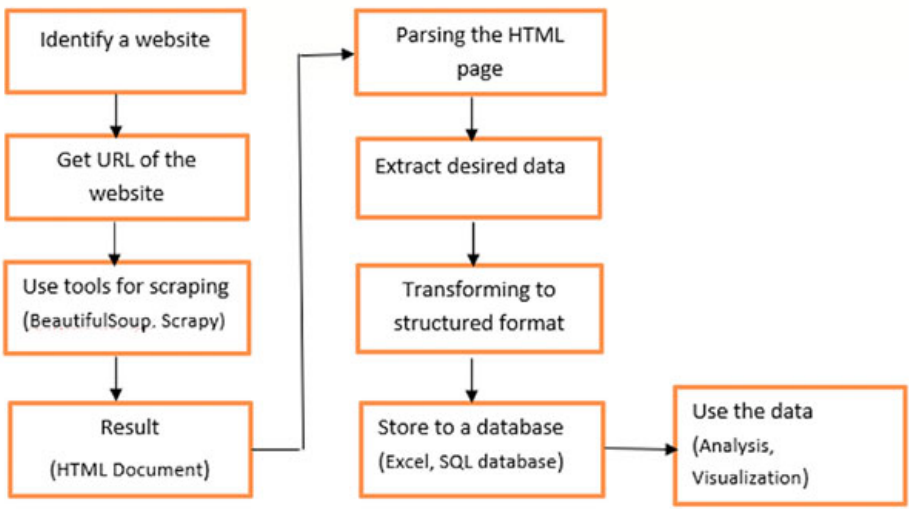


Figure 5.4: *Process flow of Web scraping process*

The complete Web scraping process can be explained in the following steps:

1. Identification of the website that is to be scrapped.
2. Obtain the URL of the website. This URL will be used further for scraping the website.
3. It is then decided what tool/programming language is to be used for scraping. In our book, we are using the Beautiful Soup library of Python. The library parses the HTML code to help in extracting data from the HTML document without writing long codes.
4. Get the HTML page of the URL using the tool or library decided in Step 3.
5. HTML data is then parsed and converted Web page content to meaningful structured data that can be understood as being in a tree-like structure of all components of an HTML document. This helps in fetching any specific data

faster.

6. Extracts the desired data from the tree-like structure generated in the preceding step.
7. The preceding data is then stored in a structured format in a spreadsheet or a database.
8. Data is then used for analysis and visualization purposes.

Challenges Of Web Scraping

Due to the dynamic nature of the Web and its contents, Web scraping is also full of challenges. A few of the challenges that one may face during scraping may arise due to the following reasons:

- Site structure may change over a period of time, so the same scraper may not always be of help.
- The technology used by the site may also change over a period of time; thus, the scraper may need some modifications as well.
- One of the important challenges that a scraper may face is the legality of the scraped data. If prohibited material is obtained, it can lead to legal issues as per the law of the country.
- Ethical issues are mostly ignored for a vested interest in the scraping of data. For example, a researcher may scrape the data for data collection purposes but may accidentally compromise on privacy of the data, especially if the data contains some persons' credentials which the user may not want to share.
- Similarly, organizations' privacy may also be unintentionally revealed, and their confidential information may get public.

Despite these challenges, since scraping gives many benefits, this field is evolving further.

Python modules used for scraping

A Web scraper is developed in the same way as any other program. It may be developed using any of the programming languages, such as Java, Python, Ruby, or JavaScript, in the framework Node.js. These languages provide libraries to use the HTML protocol to get the HTML from a Web page. In this book, we will be using the Python libraries to understand Web scraping, as Python is a very easy-to-understand language and it allows us to accomplish a complex task using a very small piece of code. The regular expressions, as seen in the first chapter, in Python make the task easier. **Beautiful Soup** Python library is a very popular library that is used for scraping. It uses a request package available in Python for enhancing its capabilities.

Another approach is the use of the framework, which takes care of each step in the

process. Python provides a framework called scrapy that provides functions for each step in the process of scraping. Some commonly used popular libraries used for scraping are Jsoup, jARVEST, Web-Harvest, and Scrapy.

Scrapy is an open-source Python framework and was originally developed for Web scraping only. Now, it supports other features like Web crawling and extracting data using APIs also. Along with its internal libraries, it allows importing external libraries like Beautiful Soup also. One of the best parts is that it supports for storing data in the cloud.

So, we can conclude that Python can be used for both methods of scraping (using programming or using a framework). Python offers a variety of libraries that one can use to scrape the Web, libraries such as Scrapy, Beautiful Soup, Requests, Urllib, and Selenium.

Legality of Web scraping

One of the very important points while scraping is regarding its legality. Is Web scraping legal?

Though there are no laws governing what can be scraped and what can not be scraped, there are certain guidelines that are laid down, and as a scraper, you must respect those guidelines. It is completely up to the scraper to respect these guidelines or ignore them completely. But, as a good programming practice, it is suggested to follow them. It definitely is a grey area, and if one scrapes a website beyond the specified area, the scraper may get into trouble.

Following guidelines can be followed before proceeding for scraping:

- Robots.txt
- Public Content
- Terms of use
- Crawl delay
- Authentication rules

Let us now see these guidelines in detail.

Robots.txt

Most of the website use robots.txt files to specify what contents scrapers can scrape and what they should not touch. As a programmer, it is your responsibility to respect these guidelines provided by the websites. This file needs to be present at the root directory of the host system, or else it is ignored by crawlers. For example:

Used: **http://example.com/robots.txt**

Ignored: **http://example.com/site/robots.txt** (robots.txt not present in the root, thus, will be ignored).

Following is a snapshot of the `robots.txt` file from the website

pythonscraping.com:

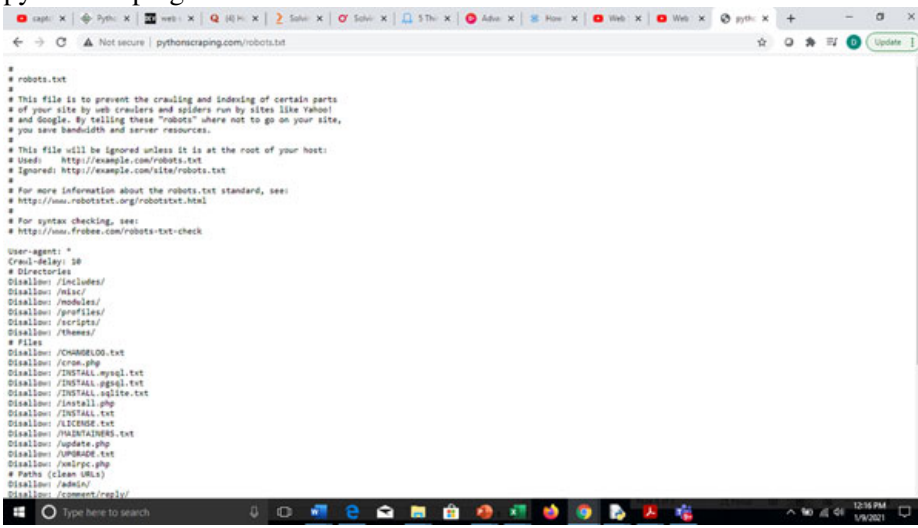


Figure 5.5: *"robots.txt" from pythonscraping.com*

Another example to demonstrate `robots.txt` from another site (myntra.com):

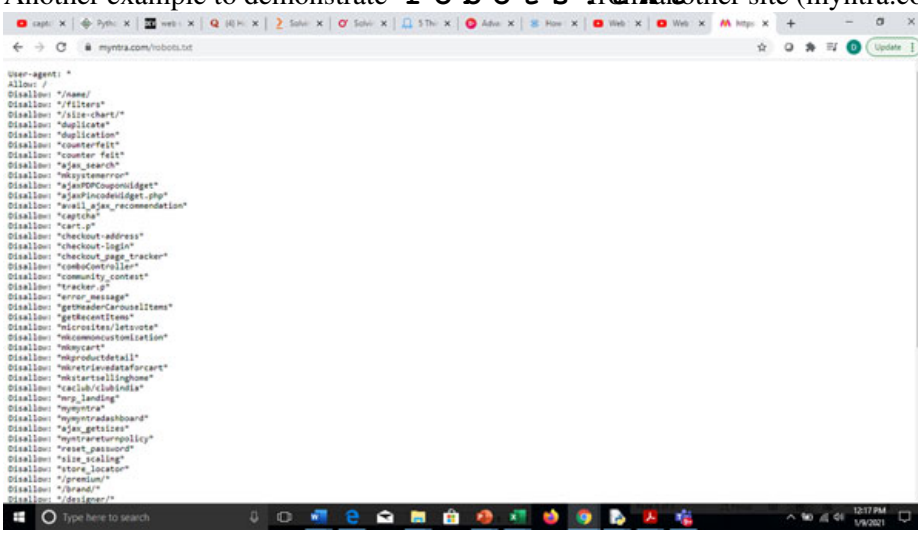


Figure 5.6: *"robots.txt" from myntra.com*

Following is a part of robots.txt from myntra.com; as you can see, it allows the crawler to scrape only the root directory but does not allow (disallow) for many other files and directories.

```
User-agent:
Allow:
Disallow: */name/
Disallow: */filters*
Disallow: */size-chart/*
Disallow: *duplicate*
```

```
Disallow: /duplication*
Disallow: /counterfeit*
Disallow: /countfeerit*
Disallow: /jax_search*
Disallow: /mksystemerror*
Disallow: /jaxPDPCouponWidget*
Disallow: /jaxPincodeWidget.php*
Disallow: /vail_ajax_recommendation*
Disallow: /captcha*
Disallow: /cart.p*
Disallow: /heckout-address*
Disallow: /heckout-login*
Disallow: /heckout_page_tracker*
Disallow: /omboController*
Disallow: /ommunity_contest*
Disallow: /racker.p*
Disallow: /rror_message*
Disallow: /getHeaderCarouselItems*
Disallow: /getRecentItems*
Disallow: /microsites/letsvote*
Disallow: /mkcommoncustomization*
Disallow: /mkmycart*
Disallow: /mkproductdetail*
Disallow: /mkretrievedataforcart*
Disallow: /mkstartsellinghome*
```

Allow specifies the pages that are ethically allowed for scraping, whereas Disallow specifies the pages that are ethically not allowed for scraping. Apart from allow and disallow, there may be other entries as well. For more information about the robots.txt standards, you are advised to go through **http://www.robotstxt.org/robotstxt.html**.

Public content

It is ideally suggested that the scrapers should do the scraping only on the public content available on the website.

Terms of use

Before starting the scraping processes, it is recommended to check the terms and conditions and privacy policy of the website from where you wish to fetch the data. If there is no such mention, then it is advised to look for the robots.txt file of the given website.

Crawl delay

Most of the websites specify crawl delay in robots.txt to specify how much delay should be maintained between crawlings. They do not want that their servers should get overloaded by crawling, thus, reducing speed for their users.

Authentication rules

Many websites allow access to their content based on authentication. They discourage crawling as they would like their website should be used by humans, not by crawlers.

A crawler must obey the guidelines to get rid of legality issues.

> ***Note:*** There is a very famous case of the legality of crawling of contents between hiQ Labs, Inc. vs LinkedIn Corp.

It is recommended to read this case to understand the legal aspect of scraping.

Data extraction and preprocessing

Gathering and getting datasets ready is one of the critical techniques in any Machine learning project or any other application where data is needed for purposes like analysis or research purpose. People accumulate the dataset through numerous approaches like databases, online repositories, APIs, survey forms, and many others. But when we want to extract any internet site data when no means of API is there, then the best alternative left is Web Scraping.

Data extraction is the process of retrieving disparate types of data from a variety of sources, many of which may be poorly organized or completely unstructured for further data processing, storage or analysis elsewhere. The term ***data collection*** is often used in place of data extraction. Automating the task of data collection/ extraction has many benefits over manual extraction of data, such as:

- Improves the accuracy of extracted data and reduces human error
- Increases speed of data collection
- Simplifies sharing of data
- Increases human productivity as the entry staff may be employed for better work instead of monotonous data entry work
- It reduces the cost of data extraction (due to automated scripts, work will be done faster, thus lesser cost)

When we talk about data, we usually think of some large datasets with a huge number of rows and columns. While that is a likely scenario, it is not always the case, and data could be in so many different forms: structured tables, images, audio files, videos, and so on. Many times the extracted data is incomplete or unstructured, or inconsistent. Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as

we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms. Without preprocessing, these errors will be carried forward and will lower the quality of the results.

Preprocessing of data is mainly to check the data quality. The quality can be checked by the following:

- **Accuracy**: To check whether the data entered is correct or not.
- **Completeness**: To check whether the data is available or not recorded.
- **Consistency**: To check whether the same data is kept in all the places that do or do not match.
- **Timeliness**: The data should be updated correctly.
- **Believability**: The data should be trustable.
- **Interpretability**: The understandability of the data.

Steps in data preprocessing include the following:

1. Data cleaning
2. Data integration
3. Data reduction
4. Data transformation

Data extraction and data preprocessing have been explained in detail in (*Chapter 6, Opinion Mining and Sentiment Analysis*) with the help of an example of sentiment analysis.

The upcoming section talks about extracting various types of data, that is, text, images, and video using Web scraping.

Handling text, image, and videos

Before you actually start writing a Web scraper, you need to visit the Web page to be scraped and give some time to understand the structure of the website. Knowing this is going to make the task of scraping super easy. Also, try to find the technologies that the target website uses. There is a Python library called "**b u i l t w**", which aims to find the technologies used by a website. This library detects the technology used by a website, such as Apache, JQuery, and WordPress. Before using this library, we need to install the library. Use **p i p i n s t a l l b u i l t w** to install it (refer to the upcoming section on how to install pip, in case you do not have it in your system).

We use the following code to find the technologies used by yahoo.com.

```
1 . i m p o r t b u i l t w i t h                    # i m p o r t
b u i l t w i t h
2 . r e s = b u i l t w i t h . p a r s e ( ' h t t p s : / / y a h o o
U R L o f w e b s i t e b u i l t w i t h . p a r s e ( )
```

```
3.print(res)                                    the
return value
4.
```

Output:

```
{'web-servers': ['Apache Traffic Server'], 'javascri
graphics': ['D3'], 'photo-galleries': ['Lightbox'],
'javascript-frameworks': ['Lightbox', 'Prototype',
'React', 'RequireJS']}
```

Some more example usage are as given as follows:

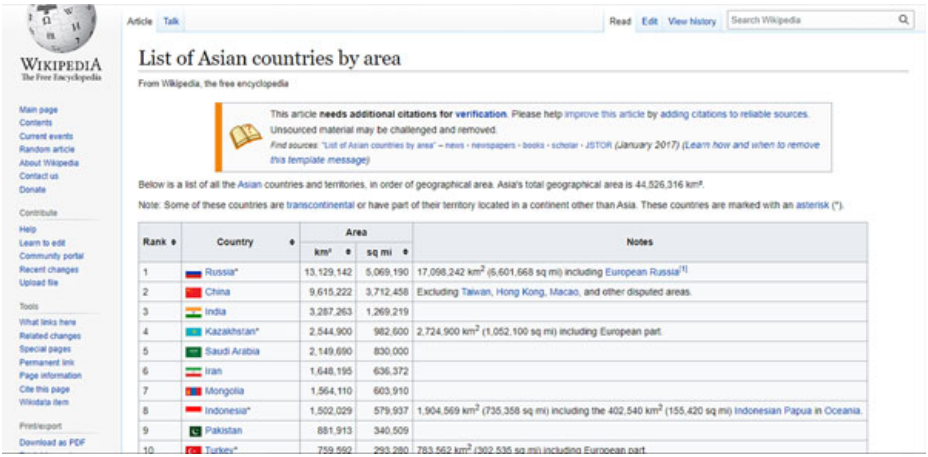
```
>>> builtwith.parse('http://wordpres
{'web-servers': ['Nginx'], 'font-scripts': ['Google
Font API'], 'ecommerce': ['WooCommerce'], 'cms':
['WordPress'], 'programming-languages': ['PHP'],
'blogs': ['PHP', 'WordPress']}
>>> builtwith.parse('http://webscrap
{'javascript-frameworks': ['Modernizr', 'jQuery'],
'web-frameworks': ['Twitter Bootstrap']}
>>> builtwith.parse('http://microsof
{'javascript-frameworks': ['Prototype', 'RequireJS',
'jQuery']}
```

In *Chapter 1, Introduction*, we saw various tools that are available for this scraping. In general, most of the simple websites (websites that are not using javascripts excessively), BeautifulSoup+requests or Scrapy can be used to write scripts that do the scraping. Scrapy can really be beneficial if you want to extract lot of data and speed up things. Scrapy is really beneficial for almost 90% of scraping jobs. However, for some other websites other tools like selenium may be beneficial.

In this chapter, we will use Beautiful Soup for demonstration purposes. In case of any other library is used, it will be mentioned explicitly.

Handling text

As we discussed in the first chapter, that BeautifulSoup library provides a very easy way to parse an HTML code of a website into a BeautifulSoup object. To understand this, let us see an example. Let us consider the following page from Wikipedia:



List of Asian countries by area

From Wikipedia, the free encyclopedia

This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed.

Below is a list of all the Asian countries and territories, in order of geographical area. Asia's total geographical area is 44,526,316 km².

Note: Some of these countries are transcontinental or have part of their territory located in a continent other than Asia. These countries are marked with an asterisk (*).

Rank	Country	Area		Notes
		km²	sq mi	
1	Russia*	13,129,142	5,069,190	17,098,242 km² (6,601,668 sq mi) including European Russia[1]
2	China	9,615,222	3,712,458	Excluding Taiwan, Hong Kong, Macao, and other disputed areas.
3	India	3,287,263	1,269,219	
4	Kazakhstan*	2,544,900	982,600	2,724,900 km² (1,052,100 sq mi) including European part.
5	Saudi Arabia	2,149,690	830,000	
6	Iran	1,648,195	636,372	
7	Mongolia	1,564,110	603,910	
8	Indonesia*	1,502,029	579,937	1,904,569 km² (735,358 sq mi) including the 402,540 km² (155,420 sq mi) Indonesian Papua in Oceania.
9	Pakistan	881,913	340,509	
10	Turkey*	759,592	293,280	783,562 km² (302,535 sq mi) including European part.

Figure 5.7: *Page from Wikipedia to be used to learn Web scraping Please refer to the link: (https://en.wikipedia.org/wiki/List_of_Asian_countries_by_area)*

Let us assume that we need to extract only the name of countries from this Web page and do not need the other text. That is, if we need a list of countries from the preceding page, it will be very difficult to copy and paste each name. In this case, we can scrap the list of countries from this page. The following code can be used to do the same:

```
1 .i m p o r t r e q u e s t s
2 .f r o m b s 4 i m p o r t B e a u t i f u l S o u p
3 .
4 .U R L = " h t t p s : / / e n . w i k i p e d i a . o r g / w i k i
L i s t _ o f _ A s i a n _ c o u n t r i e s _ b y _ a r e a "
5 .p a g e = r e q u e s t s . g e t ( U R L )
6 .
7 .s o u p = B e a u t i f u l S o u p ( p a g e . t e x t , ' h t m l . p a r s e r
8 .t a b l e = s o u p . f i n d ( " t a b l e " , c l a s s _ = " w i k
9 .b o d y = t a b l e . f i n d ( " t b o d y " )
1 0 .
1 1 .c o u n t r i e s = [ ]
1 2 .
1 3 .f o r p a r e n t i n b o d y . f i n d _ a l l ( " t d " ) :
1 4 .   h a s _ i m g = p a r e n t . f i n d ( " i m g " )
1 5 .   i f h a s _ i m g :
1 6 .     c o u n t r i e s . a p p e n d ( p a r e n t . f i n d ( "
1 7 .
1 8 .p r i n t ( c o u n t r i e s )
```

Handling images

Many times we may need to scrape images (maybe for preparing the dataset or for any other purpose); we can scrap the images from any Web page and store them in

our hard drive. We will now see how Python (BeautifulSoup) can be used to scrape the images using the following code. We are using Web page **https://rubikscode.net/** to extract all images.

Example 1:

```
1 .import requests
2 .from bs4 import BeautifulSoup
3 .import os
4 .
5 .url = 'https://rubikscode.net/'
6 .
7 .ur = requests.get(url)
8 .
9 .soup = BeautifulSoup(ur.text, 'html.parser')
10 .
11 .images = soup.find_all('img')
12 .
13 .for image in images:
14 .    print(image['src'])
15 .
```

To do so, we will import **Beautiful Soup** from the **requests** library. We are also importing "os" module as we need to store the images in the hard drive. "os" module is used whenever the code needs to interact with the underlying operating system. In Line 5, we are storing the URL of the website from which images need to be scraped into the variable "url". Then in Line 7, we pass this "url" to the `get()` method of "requests", to connect to and retrieve information from the given server using a given URL. This information is stored in the variable "ur". At this stage, if you will get lots of information in the form of HTML text. Line 9 uses "BeautifulSoup" to create a parse tree from the page source to extract data in a hierarchical and more readable manner, which is stored in "soup" variable (Conventionally, we use the "soup" variable name, but any name can be used). At this stage, if you print the value of "soup" variable, you will see the entire Web page with HTML tags. Now, we want only the images from the page, and we have seen that each image link is specified in `<img>` tag. So, using find_all('img') in Line 11, we find all image links from the soup object. We now iterate the list of links stored in "images" using a for loop in Line 13 and print each link in Line 14. You will get the output as follows:

```
https://rubikscode.net/wp-content/uploads/2020/02/RC-logo-for-light-bg.png
https://i2.wp.com/rubikscode.net/wp-content/uploads/2021/04/AdobeStock_396897385-1-scaled.jpeg?w=1080&ssl=1
https://i2.wp.com/rubikscode.net/wp-content/uploads/2021/05/Untitled-design.png?w=1080&ssl=1
https://i2.wp.com/rubikscode.net/wp-content/uploads/2021/04/Colorful-Abstract-Design-Book-Cover.png?w=1080&ssl=1
https://i2.wp.com/rubikscode.net/wp-content/uploads/2020/01/imageedit_25_5570116102.png?w=1080&ssl=1
https://i1.wp.com/rubikscode.net/wp-content/uploads/2020/01/imageedit_20_6143676424.png?w=1080&ssl=1
https://i1.wp.com/rubikscode.net/wp-content/uploads/2020/01/imageedit_15_4262047064.png?w=1080&ssl=1
https://i0.wp.com/rubikscode.net/wp-content/uploads/2021/10/Featured-45.png?resize=400%2C250&ssl=1
https://i2.wp.com/rubikscode.net/wp-content/uploads/2020/11/Featured-43.png?resize=400%2C250&ssl=1
https://i0.wp.com/rubikscode.net/wp-content/uploads/2021/07/Featured-4.png?resize=400%2C250&ssl=1
```

Figure 5.8: *Output of Example 1 that returns links of images in the scraped page*

If you paste any of these links on the Web browser, you will be able to see the extracted image.

Note: get() method of the Requests library is used for making an HTTP request to a specific URL.

"BeautifulSoup" is a Python library to pull data from HTML, XML, and other markup languages.

Now, once we have got the images, we can store these images on our hard drive also. In the following code, we will try to do the same.

```
1 .# # s t o r i m g g i s n t h e c u r r e d i t r e c t o r y
2 .i m p o r t e q u e s t s
3 .f r o m s 4 i m p o B e a u t i f u l S o u p
4 .i m p o o t s
5 .
6 .u r l = ' h t t p s : / / r u b i k s c o d e . n e t / '
7 .r = r e q u e s t s . g e t ( u r l )
8 .s o u p = B e a u t i f u l S o u p h t m l . p x a r s e r ' )
9 .i m a g e s = s o u p . f i n d _ a l l ( ' i m g ' )
1 0 .
1 1 .f l = 0
1 2 .
1 3 .f o r i m a g i e n i m a g e s :
1 4 .    l i n k = i m a g e [ ' s r c ' ]
1 5 .    f l = i n t ( f l ) + 1
1 6 .    w i t h o p e n ( ' f i l e ' + s t r ( f l ) + ' s f j : p g ' ,
1 7 .        i n = r e q u e s t s . g e t ( l i n k )
1 8 .        f . w r i t e ( i n . c o n t e n t )
1 9 .
```

The initial part of scraping images remains the same. Images are written to the disk in line number 13 to 19. We have already discussed the use of Lines 13 and 14 in the previous paragraph while scraping the image. In Line 16, the open method is used to open/create a file where we need to pass two arguments, filename and the mode in which the file is to be created. Since we would like each image to be stored in a separate file (and have .jpg extension), we are specifying a variable filename as explained as follows. We are giving filenames as file1, file2, ….file *n*, where *n* is the total number of images extracted. We are specifying file mode as "wb" as we would be interested in storing the file content as binary data. In Line 17, we are sending the request to get the link which is stored in the variable "in". In Line 18, we write the contents of the link stored in "in" using fwrite (in.content).

In Line 16, `' f i l e ' + s t r ( f l ) + ' . j p g '` are adding three strings, "file", variable part in the filename (1,2,3…*n*) and ".jpg" to create the filename as file1.jpg, file2.jpg….filen.jpg. Line 11 initialises the value of variable part "fl" as

zero, and Line 15 is incrementing it by one in each iteration. All these files will be stored in the current directory (as shown in the red encircle in *figure 5.9*):

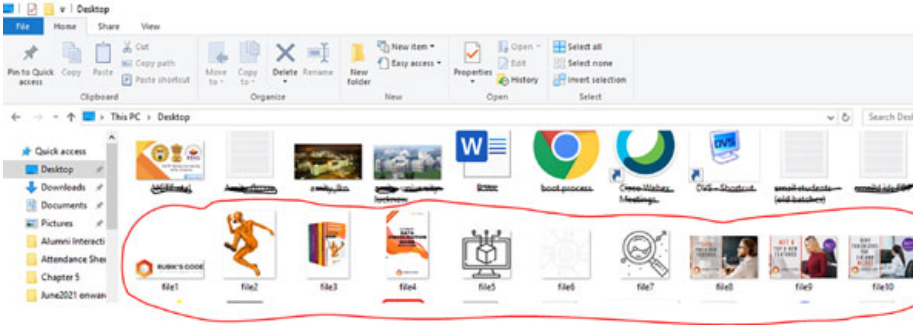


Figure 5.9: *Downloaded images in the form of image(jpg) files (file1, 2,...10)*

Let us take one more example that focuses to scrape the image stored inside a table. Let us go back to the example in the previous section, where we extracted the names of countries.

Exercise: Try to extract only the flag of countries from the Web page in *figure 5.7*, excluding the other text. This can easily be done through image scraping. This is for your exercise.

Example 2:

One more example to demonstrate extracting images from the Web page:

```
1.# Extraction image from a web page
2.from urllib.request import urlopen
3.from bs4 import BeautifulSoup
4.import re
5.
6.html = urlopen('https://en.wikipedia.org/wiki/
Peter_Jeffrey_(RAAF_officer)')
7.bs = BeautifulSoup(html, 'html.parser')
8.images = bs.find_all('img',
{'src':re.compile('.jpg')})
9.for image in images:
10.    print(image['src'] + '\n')
```

Output:

```
//upload.wikimedia.org/wikipedia/commons/thumb/1/1a/
NlaJeffrey1942-43.jpg/220px-NlaJeffrey1942-43.jpg
//upload.wikimedia.org/wikipedia/commons/thumb/
c5/008315JeffreyTurnbull1941.jpg/260px-008315JeffreyTurnbull1941.jpg
//upload.wikimedia.org/wikipedia/commons/thumb/
ea/021807CameronJeffrey1941.jpg/
//upload.wikimedia.org/wikipedia/commons/thumb/
AC0072JeffreyTruscottKittyhawks1942
```

```
AC0072JeffreyTruscottKittyhawks1942
//upload.wikimedia.org/wikipedia/co
VIC1689Jeffrey1945.jpg/280px-VIC168
```

The code returns the link of images present on the Web page used in line number 8. These images may be downloaded and used as per requirement. The code may not work for each and every page from which you are trying to fetch the images in the exact code as in the beginning, and we discussed that each Web page is different, and before writing code, it is important to understand the structure of the Web page. This code serves as a basic guideline for how to connect and fetch the image using Beautiful Soup. Minor modifications may be required as per the structure.

Note: Go through the BeautifulSoup documentation at https://www.crummy.com/software/BeautifulSoup/bs4/doc/ to learn the functionality of this library in detail.

Also, go through the commonly used HTML tags for a better understanding of a Web page.

Extracting videos from a Web page

To understand how videos can be scraped from a Web page, we will discuss scraping video and other video-related information from one of the biggest video-sharing websites that is YouTube. It contains the most popular videos and other related information to each video (number of likes, number of subscribers, comments and so on). We will use BeautifulSoup and requests libraries of Python.

The examples considered in this section are to understand the basics; however, as YouTube keeps on changing its code, this code may also need changes from time to time. As of now, all the codes discussed are giving the correct results. A more reliable method can be developed by the use of APIs for extracting data.

In this section also, for Web scraping, we will use **requests and BeautifulSoup** Modules in Python. The **requests** library is an integral part of Python for making HTTP requests to a specified URL. Whether it be REST APIs or Web Scraping, requests are must be learned for proceeding further with these technologies. When one makes a request to a URL, it returns a response. Python requests provide inbuilt functionalities for managing both the request and response.

Let us understand the concept using the following example:

```
1.fromurllib.requestimporturlopen
2.frombs4importBeautifulSoup
3.importre
4.
5.html=urlopen('https://www.youtube.
v=-5B858LWJD0')
6.
```

```
7.bs=BeautifulSoup(html,'html.parser')
8.
9.# sample youtube video url
10.video_url="https://www.youtube.com/
v=-5B858LWJD0"
11.# init an HTML Session
12.# get the html content
13.session=requests.Session()
14.response=session.get(video_url)
15.bs.find_all("meta")
```

After preliminary statements, the use of which we have already seen, line number 10 indicates the URL of the page from where the video is to be extracted. Before actually scraping the video, let us understand the Web page, for with we try to understand metadata about the returned HTML page. **<meta>** tag in line number 15 is used to get this information. After executing the code, we get the result as follows:

Output:

```
<meta content="1280" property="og:video:wid
<meta content="720" property="og:video:heig
<meta content="YouTube" property="al:android
>,
<meta content="com.google.android.you
property="al:android:package"/>,
<meta content="best diet plan for kidney patients"
property="og:video:tag"/>,
<meta content="best diet for kidney patients"
property="og:video:tag"/>,
<meta content="best diet plan for kidney failure
patients" property="og:video:tag"/>,
<meta content="best diet for kidney failure patients
property="og:video:tag"/>,
<meta content="best diet plan for kidney disease
patients" property="og:video:tag"/>,
<meta content="рдХрд рдбрд╝рдиреА рд░реЛрдЧ
рд рд рдб diet chart" property="og:video:tag
<meta content="diet plan for kidney patients"
property="og:video:tag"/>,
<meta content="diet for kidney patients"
property="og:video:tag"/>,
<meta content="diet plan for kidney failure patients
property="og:video:tag"/>,
<meta content="diet for kidney failure patients"
property="og:video:tag"/>,
```

```
<meta content="diet plan for kidney disease patients"
property="og:video:tag"/>,
<meta content="sanjivani" property="og:vide
>,
<meta content="puru dhawan" property="og:vid
>,
<meta content="kidney treatment"
property="og:video:tag"/>,
<meta content="kidney disease treatment"
property="og:video:tag"/>,
<meta content="kidney disease" property="og:vid
>,
<meta content="87741124305" property="fb:app_i
<meta content="player" name="twitter:card"/>
<meta content="@youtube" name="twitter:site"/
<meta content="https://www.youtube.co
v=-5B858LWaI0" name="twitter:url"/>,
<meta content="Diet Plan for Kidney Patient | By
Dr.Puru Dhawan" name="twitter:title"/>,
<meta content="This video Dr.Puru Dhawan provide
information on diet to be taken during kidney
failure. Sanjivani contact details: For Appointme
call 9811744..." name="twitter:description
<meta content="https://i.ytimg.com/vi
maxresdefault.jpg"
.....
......
<meta content="1280" name="twitter:player:w
<meta content="720" name="twitter:player:he
<meta content="Diet Plan for Kidney Patient | By
Dr.Puru Dhawan" itemprop="name"/>,
<meta content="This video Dr.Puru Dhawan provide
information on diet to be taken during kidney
failure. Sanjivani contact details: For Appointme
call 9811744..." itemprop="description"/>,
.....
```

We can see the output returns entire metadata information as can be seen in the following line:

```
<meta content="Diet Plan for Kidney Patient | By
Dr.Puru Dhawan" itemprop="name"/
```

Notice that the `itemprop` attribute specifies the properties of the Web page, the `name` specifies that we are interested in the name of the Web page, and from the `content` we get the content of `name` property. Extracting this value will return the name of the Web page. Add the following line to the code, and we will get

name/title of the video that we are scraping.

```
bs.find("meta",prop="name")["content
```

Output:

```
'BestDietPlanforKidneyPatientsbyDr.PuruDhawan'
```

So, we have successfully managed to fetch the title of the video. Similarly, using other properties, you will be able to extract everything you want from that Web page.

Using the following command, we can extract the number of views using the `interactionProperty`. Using the following command, we get the desired result.

```
soup.find("meta",prop="interactionCou
['content']
```

Output:

```
'84841'
```

So, the number of views of this video is "84841".

Exercise:

The following command can be used to get a description of the video. Try to implement this command to scrape the description of the video.

```
bs.find("meta",prop="description")['
```

This way, you will be able to extract everything you want from that Web page.

We can see the complete code in one place as follows:

```
1.# sample youtube video url
2.video_url="https://www.youtube.com/w
v=-5B858LWJD0"
3.# initiate HTML Session
4.# get the html content
5.session=requests.Session()
6.response=session.get(video_url)
7.
8.# get meta data of the youtube video
9.bs.find_all("meta")
10.
11.# get title of video
12.name=bs.find(item=metapar"o,p="name")["co
13.
14.# get number of views
15.count=bs.find(item="metapar"o,p="interacti
['content']
16.
17.# get description
```

```
18.desc=bs.find(itemprop="description")['content']
19.
20.print("Title of the video:",name,"\nNumber of views are:",count,"\nDescription of video:",desc)
21.
```

Output:

```
Title of the video:    Best Platform for Kidney Patients by Dr. Puru Dhawan
Number of views are:    85258
Description of video:    This video, Dr. Puru Dhawan provides information about diet to be taken during kidney failure. Sanjivani contact details: For Appointment call 9811744 9…
```

Now, we will see one more example to demonstrate storing these properties in a dictionary instead of separate variables for further use.

Example:

```
1.# importing the libraries
2.from urllib.request import urlopen
3.from bs4 import BeautifulSoup
4.import re
5.
6.# Defining an empty dictionary to which the properties shall be stored
7.result={}
8.
9.html=urlopen('https://www.youtube.com/watch?v=-5B858LWJD0')
10.
11.# Parsing HTML code
12.soup=BeautifulSoup(html,'html.parser')
13.
14.# sample youtube video url
15.video_url="https://www.youtube.com/watch?v=-5B858LWJD0"
16.session=requests.Session()
17.response=session.get(video_url)
18.
19.# video title
20.result["title"]=soup.find("meta",itemprop="name")['content']
```

```
# video views (convert to integer)
result["views"] = result["views"] = soup.find("meta", itemprop="interactionCou
['content']

# video description
result["description"] = soup.find("meta",
itemprop="description")['content']

# date published
result["date_published"] = soup.find("meta",
itemprop="datePublished")['content']

# get the duration of the video in minutes and seconds
result["duration"] = soup.find("meta",
itemprop="duration")['content']

# print results as, sum. Here is the key following
traverse through the dictionary and print key and
corresponding values
for i in result:
    print(i,":", result[i])
    print("\n")
```

Output:

```
title: Best Diet Plan for Kidney Patients By Dr.
Puru Dhawan
views: 85259
description: In this video, Dr. Puru Dhawan provides
information about diet to be taken during kidney
failure. Sanjivani contact details: For Appointment
call 98117449…
date_published: 2021-06-10
duration: PT19M5S
```

In this example, the values of the parameters name("`itemprop`") is stored as keys in the dictionary "result" along with its value ("`content`") as values of the dictionary. Later the dictionary is traversed using a for loop to display all the "`key : value`" pairs from the dictionary "`result`".

Scraping dynamic websites

Dynamic websites are websites that change the content or layout with every request. A dynamic website is a type of website that can update or load content

after the initial **HTML** load. It provides information that is updated frequently and may need to access a database. Information may be obtained based on login and password. In this section, we will learn to scrape dynamic Web page based on login, extracting videos, images, and pages that deal with captcha.

Scraping websites with login

To start scraping pages with login, the two elements needed to post a response to a site and login are as follows:

1. The name of the fields you want to push data to
2. The url of the page the data actually posts to on the backend

We use the test website "**www.https://www.chess.com**", to understand how to login through a scraper using the internal variables that are placeholders for login name and password, as shown in the following figure:

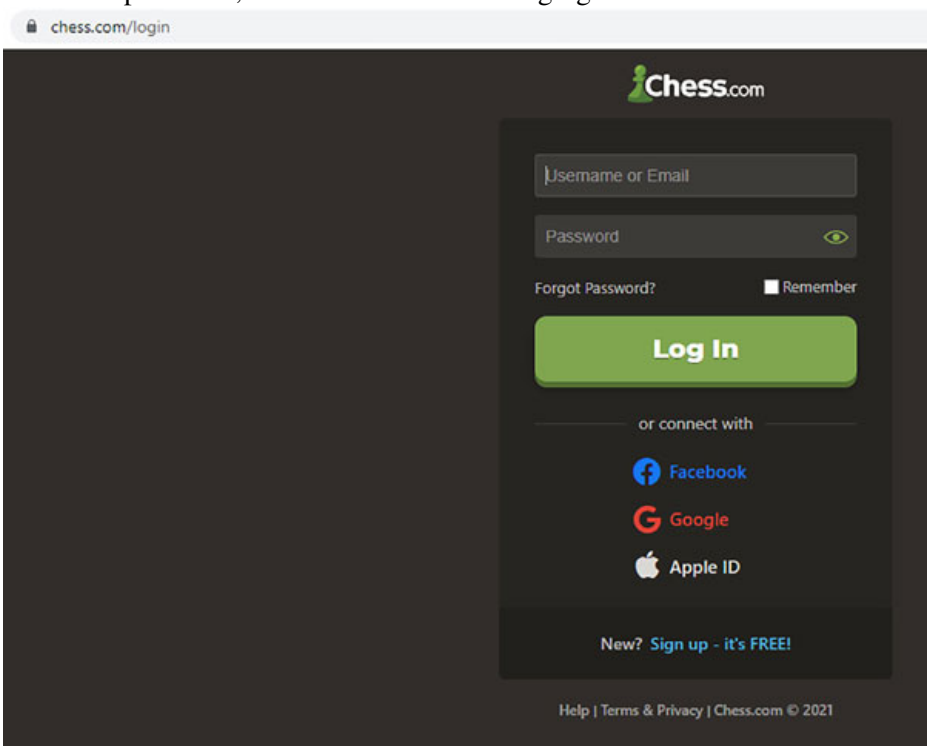


Figure 5.10: *Web page with login for scraping*

After opening the page, go to the username field and press right->inspect. You will get the following page as shown in *figure 5.11:*

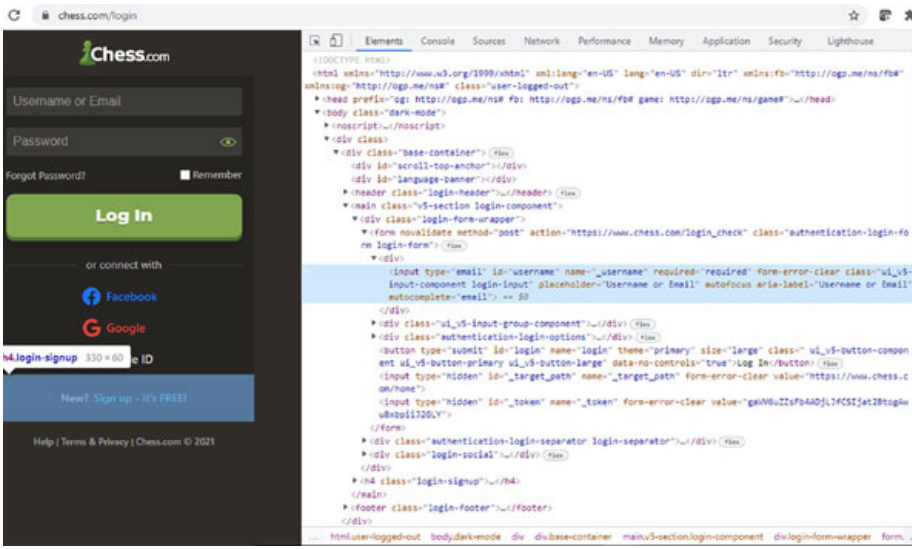


Figure 5.11: *Output of inspect window*

In the highlighted portion, which is the code related to the input username input box, notice the placeholder for the user as "**n a m e = ' _ u s e**". Similarly, when click on inspect by right clicking in the password filed field, you will get the placeholder for the password as "**n a m e = ' _ p a s s w o r d '**

The following part of the code helps to understand how to login to a Web page programmatically while creating the Web scraper.

```
1 . f r o m b s 4 i m p o r t B e a u t i f u l S o u p
2 . i m p o r t r e q u e s t s
3 .
4 . # S t a r t t h e s e s s i o n
5 . s e s s i o n = r e q u e s t s . S e s s i o n ( )
6 .
7 . # C r e a t e t h e p a y l o a d
8 . p a y l o a d = { ' _ u s e r n a m e ' : ' [ Y O U R _ U S E R N A M
9 .                 ' _ p a s s w o r d ' : ' [ Y O U R _ P A S S W O R D
1 0 .                 }
1 1 .
1 2 . # P o s t t h e p a y l o a d t o t h e s i t e t o l o g i n
1 3 . s = s e s s i o n . p o s t ( " h t t p s : / / w w w . c h e s
l o g i n _ c h e c k " , d a t a = p a y l o a d )
1 4 .
1 5 . # N a v i g a t e t o t h e n e x t p a g e a n d s c r a p e t h e d a t a
1 6 . s = s e s s i o n . g e t ( ' h t t p s : / / w w w . c h e s s
1 7 .
1 8 . s o u p = B e a u t i f u l S o u p ( s . t e x t , ' h t m l . p a r s e r ' )
1 9 . s o u p . f i n d ( ' i m g ' ) [ ' s r c ' ]
```

Output:

https://images.chesscomfiles.com/uploads/v1/news/1082489.f346be64.630x354o.7073d3433a6a.png

In this code, Line 5 starts a session and is followed by Lines 8 and 9, where a dictionary called payload is created that contains "`_username`" and "`_password`" as keys and the values contain the actual username and password. In this example, the username is "`YOUR_USERNAME`" and the password is "YOUR_PASSWORD". In Line 15, these values (data=payload) are posted to the specified Web page **https://www.chess.com/login_check** using the `session.post()` method. Line 16 gets the data from the Web page mentioned as an argument (note that we have already logged in to the page in Line 15); Line 18 parses the page contents using BeautifulSoup() method. Now onwards, scraping can be done in the way as explained in *Chapter 3, Prominent Applications with Web Mining*. In our example, we are getting the output as a link to the image as follows:

https://images.chesscomfiles.com/uploads/ad_image/10149.ec98eaaa.png

If you post this link on the browser, you will get the image that is part of the Web page (scraped image).

Dealing with CAPTCHA

The term **Completely Automated Public Turing test to tell Computers and Humans Apart** (**CAPTCHA)** was introduced in the year 2000 at Carnegie Mellon University. It is quite often that you are asked to solve this before accessing certain websites. Many a time, solving these CAPTCHAs becomes annoying. But they are used to prevent bots from scraping Web pages. The sole purpose is to protect data. CAPTCHA is used to test whether a visitor is an actual human or an automated/programmed bot. But the question is, what is the need to make this distinction? The answer is that these two entities have an entirely different purposes to visit the Web page. In general, when human visits a website, they will use it for some purpose, such as banking, shopping and so on, and in the meanwhile, they may click some ads as well. All such activities are in favour of the Web page and can help in creating revenue for the Web page. But, the bots are usually created with ill intentions and do not help in generating revenue, instead many times they may harm the site with malware or DDOS attack. If the site is able to differentiate between humans and bots, it can block bots and can cut down the risk of attacks.

Captcha is a visual interface to stop automated computer programs, also known as bots and spiders, from gaining access to the website. Manual Blum, a computer science professor at Carnegie Mellon University, along with his team Luis Von Ahn Ben and others, came up with the first CAPTCHA, which is an acronym for "Completely Automated Public Turing test to tell Computers and Humans apart".

Based on the level of complexity, different types of CAPTCHA are being used. Some of the most common ones are listed as follows:

- **Character Recognition**

  In this type of captcha, an image consisting of characters and numbers that are blurred or distorted is presented. This image is easily readable by a human but could pose challenges for a bot to solve it easily. Due to good libraries for image processing, now it is possible to write a computer program that can read the characters and numbers in this image.

- **Audio Recognition**

  This type of captcha is of help to visually impaired people. In this, the captcha players a distorted recording, and the user is required to type what they have heard. Though, now such a type of captcha is becoming less popular due to evolving audio recognition tools that provide a good level of accuracy.

- **Math Problems**

  This type of bot presents a simple math problem like “7+4= “. Such problems can be easily comprehended and solved by humans, but difficult for a bot to parse and answer them. Such problems can also be presented as word problems like “7 plus 4= “ “, so any standard solution may not work for both methods.

- **Image Recognition**

  This is a more secure and reliable type of captcha. Image-based captcha was developed to replace text-based captcha. These captchas use recognizable graphical elements like photos of animals and scenes, and the user is required to identify a specific animal or some element in the scenes.

- **Social Media Logins**

  Some Web pages ask the user to log on with a social media account in order to access a website. Though this is more secure as bots can not accomplish this task as only the user knows the login details of his/her social media account; since users hesitate to login to their social media account, thus, this is not a very popular method.

  CAPTCHA is developed by well-funded and highly experienced developers, and their aim is to develop CAPTCHA in a way to be as difficult as possible for non-humans to access places on the website that should not be. In case a captcha is programmatically solved, it is modified by the developers so as to increase the level of difficulty.

Captcha can be solved by following methods:

- Solve it by writing programs: CAPTCHAs can be solved by writing computer codes to decode the CAPTCHA
- Pay people to do it manually for you:
- Pay people who can solve it for you

- Use available tools for the purpose of bypassing captchas

Web scraping will require to bypass or solve the captchas through the program. In this book, we will see how Python can be used to bypass captcha before scraping the page. Basic ethics that were followed for scraping any page will apply here too.

Case study: Implementing Web scraping to develop a scraper for finding the latest news

After having learned about Web scraping and how to scrape different types of data (text, images, and videos) and other related things, we now move ahead to see an example case study.

Our problem statement for the case study is "In your regular routine, you like to watch top news related to any specific category without opening multiple sites". As a solution to your problem can be solved by using Web scraping techniques. You need a Web scraper that finds news based on categories such as BUSINESS, SCIENCE, TECH, WORLD, ENTERTAINMENT and so on and displays the top 10 news from the section based on users' choice.

In this case study, the code will display 10 sections from which the user can choose a category. The top 10 news/results from the mentioned website are displayed. In this code, we are taking input of "*category*" from the user and the name of website is hardcoded in the code.

Code:

```
1.frombs4importBeautifulSoup
2.importrequests
3.fromtabulateimporttabulate
4.
5.urls={
6.    1:"https://www.indiatoday.in/tr
7.    2:"https://www.indiatoday.in/bu
8.    3:"https://www.indiatoday.in/sc
9.    4:"https://www.indiatoday.in/wo
10.    5:"https://www.indiatoday.in/t
11.    6:"https://indianexpress.com/s
entertainment/",
12.    7:"https://indianexpress.com/s
13.    8:"https://www.bankbazaar.com/
price-india.html",
14.    9:"https://www.bankbazaar.com/
price-india.html",
15.    10:"https://www.timeanddate.co
16.}
17.
18.categories
```

```
    1: "TRENDING NEWS",
    2: "BUSINESS",
    3: "SCIENCE",
    4: "WORLD",
    5: "TECH",
    6: "ENTERTAINMENT",
    7: "SPORTS",
    8: "PETROL PRICES",
    9: "DIESEL PRICES",
    10: "WEATHER"
}

def main():

    print("=========================
    print("=====NEWS SCRAPER=========
    print("=========================

    for _, index in enumerate(categories
        print(index, ":", categories[i

    print("Enter a number (1 - 10) or 0 to EXIT:")
    category = int(input())

    news_table = []

    print()
    print("SHOWING NEWS FOR CATEGORY:
", categories[category])
    print()

    if category > 0 and category <= 10:
        page = requests.get(urls[catego
        soup = BeautifulSoup(page.conte
"html.parser")

        if category <= 4:
            news = soup.find_all("div",
class_="catagory-listing")
            for block in news:
                title = block.find("p").te
                link = block.find("a").get
                news_table.append([titl
```

```
        if category == 5:
            news_ul = soup.find("ul", class_="it
listing")
            news_li = news_ul.find_all("l
            for block in news_li:
                title = block.get(‹title›)
                link = block.find("a").get
                news_table.append([titl

        if category == 6:
            news_articles = soup.find_all("d
class_="articles")
            for block in news_articles:
                title = block.find("p").te
                link = block.find("a").get
news_table.append([title, link])

        if category == 7:
            news_articles = soup.find_all("d
class_="articles")
            for block in news_articles:
                title = block.find("p").te
                link = block.find("a").get
news_table.append([title, link])

        if category == 8:
            main = soup.find(id="grey-bt
            trs = main.find_all("tr")
            total = len(trs)
            table = []
            table.append(["CITY", "PRI
            for i in range(1, total):
                tds = trs[i].find_all("td
                city = tds[0].find(‹a›).te
                price = tds[1].text
table.append([city, price])
print(tabulate(table, headers=›first
            print()

        if category == 9:
```

```
97.          main=soup.find(id="grey-bt
98.          trs=main.find_all("tr")
99.          tota=len(trs)
100.          tabl=e[]
101.          table.append(["CITY","PRI
102.          fori inrange(1,tal):
103.              tds=trs[i].find_all("t
104.              city=tds[0].find(‹a›).t
105.              pric=etds[1].text
106.
table.append([city,price])
107.
print(tabulate(table,headers=›first
108.          print()
109.
110.      ifcatego=ry10:
111.          main=soup.find("table",cla
fwtb-wzteb rvaa-m")
112.          trs=main.find_all("tr")
113.          tota=len(trs)
114.          tabl=e[]
115.          fori inrange(1,tal):
116.              tds=trs[i].find_all("t
117.
118.              city=tds[0].find(‹a›).t
119.              upd=tds[1].text
120.              comn=t
tds[2].find(‹img›).get(‹title›)
121.              temp
tds[3].text
122.
table.append([city,temp,comnt,upd])
123.
124.              city=tds[4].find(‹a›).t
125.              upd=tds[5].text
126.              comn=t
tds[6].find(‹img›).get(‹title›)
127.              temptds[7].text
128.
table.append([city,temp,comnt,upd])
129.
130.              city=tds[8].find(‹a›).t
131.              upd=tds[9].text
132.              comn=t
```

```
tds[10].find(<img>).get(<title>)
            temp=tds[11].text

table.append([city,temp,comnt,upd])

        cities=[]

        for city in table:
            cities.append(city[0])

        cities=sorted(cities)

        final_table=[]

final_table.append(["CITY","TEMP","

        for city in cities:
            for city_block in table:
                if city==city_block[0]

final_table.append(city_block)


print(tabulate(final_table,headers=
        print()

  if category==7:
     for news_block in news_table:

print("-------------------------------
----------")
        print("Title:",news_block
        print()
        print("Link:",news_block[

print("-------------------------------
----------")
        print()

  print(<Source>+sources[category])
  print()

main()
```

Output:

After we run the program, the following options are displayed. Selecting a specific option gives top news related to the category of the option provided.

```
=====================================
======== NEWS SCRAPE ==========
=====================================
1 : TRENDING NEWS
2 : BUSINESS
3 : SCIENCE
4 : WORLD
5 : TECH
6 : ENTERTAINMENT
7 : SPORTS
8 : PETROL PRICES
9 : DIESEL PRICES
10: WEATHER
Enter choice(1-10) or 0 to EXIT :
5
```

On making a selection as 5, we get the following results that are top news of category 5, that is, "TECH" for the day when you run the scraper.

```
-------------------------------------
Title: Tencent announces layoffs for first time in 10
years, fires nearly 5500 employees
Link: /technology/news/story/tencent-
for-first-time-in-10-years-fires-ne
employees-1989097-2022-08-17
-------------------------------------
Title: Should you buy iPhone 13 or wait till next month
for iPhone 14?
Link: /technology/talking-points/stor
iphone-13-or-wait-for-iphone-14-198
-------------------------------------
Title: ACT Fibernet Delhi is offering broadband plans
with more than 150 mbps speed with additional benefits
Link: /technology/news/story/act-fibe
offering-broadband-plans-with-more-
with-additional-benefits-1989096-20
-------------------------------------
Title: WhatsApp will soon give users the option to
recover deleted messages
Link: /technology/news/story/whatsapp
users-the-option-to-recover-deleted
messages-1989045-2022-08-17
-------------------------------------
```

```
Title: Reliance Jio and Viacom18 in final talks for multi-billion 5G related deals
Link: https://www.indiatoday.in/technology/news/story/jio-and-viacom18-in-final-talks-for-multi-billion-5g-related-deals-1989160-2022-08-17
--------------------------------------------------
Title: Motorola launches Moto G62 tablet in India
Link: https://www.indiatoday.in/technology/news/story/motorola-moto-g62-tablet-in-india-1989024-2022-08-17
--------------------------------------------------
Title: Vivo V25 Pro launched in India, price starts from Rs 35,999
Link: https://www.indiatoday.in/technology/news/story/vivo-v25-pro-india-price-starts-from-rs-35-999-1988870-2022-08-17
--------------------------------------------------
Title: 5G launch in India will happen soon, speed will be 10 times faster than 4G: PM Modi
Link: https://www.indiatoday.in/technology/news/story/5g-launch-happen-soon-speed-will-be-10-times-faster-than-4g-pm-modi-1988858-2022-08-17
--------------------------------------------------
Title: WhatsApp standalone app for Windows users, here's how to download
Link: https://www.indiatoday.in/technology/news/story/whatsapp-standalone-app-for-windows-users-here-s-how-to-download-it-from-1988936-2022-08-17
--------------------------------------------------
Title: Scammers WhatsApp to send fake job offers, don't fall for your reply, accept
Link: https://www.indiatoday.in/technology/news/story/scammers-whatsapp-to-send-fake-job-offers-do-not-fall-for-your-reply-accept-1988896-2022-08-17
--------------------------------------------------
Title: Man charged Rs 3,000 for 45 km Uber ride from Delhi Airport to Noida
Link: https://www.indiatoday.in/technology/news/story/man-travels-airport-to-noida-in-uber-gets-slapped-with-bill-for-45-km-ride-1988858-2022-08-
```

Source: **https://www.indiatoday.in/technology**

Similarly, by giving other options, you can get news related to it.

Additional bit

Since most of the examples have been implemented using Beautiful Soup. This section discusses the method of installing “pip” and “BeautifulSoup” packages.

"pip" is the installer that is used to install other packages in Python. In this section, we first discuss the installation of "pip", followed by the installation of "BeautifulSoup". All examples, however, have been implemented in Google Colab, which does not need to install any library explicitly, but this knowledge will be helpful for those who use other interpreters of Python.

Once Python is installed on your system, you should download pip. Pip is a package management system that is written using Python and is used to install and manage packages in Python. Pip will be used whenever additional libraries are needed. Like other software, we will first download it and then extract it before it can be used for software management.

Downloading pip.

1. Search for **get pip py** using google, and the first result will be shown as follows:

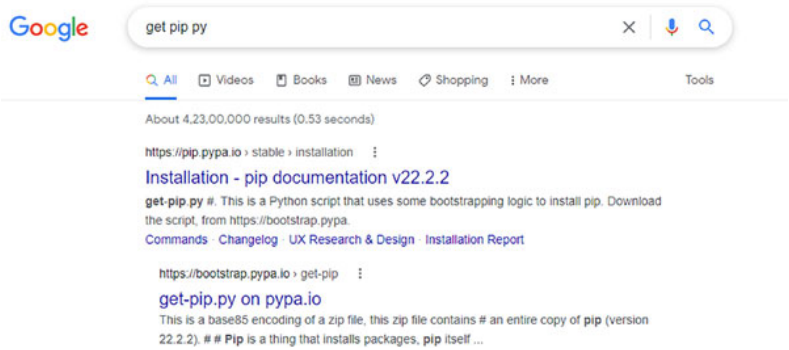


Figure 5.12: *Result of get pip py*

2. Right-click on the link and click on **"Save Link as"**.
3. Save the given link on your desktop.
4. Open the command prompt (*Windows key+r*), type cmd and hit enter
5. You will get a command prompt, as shown in the following figure:

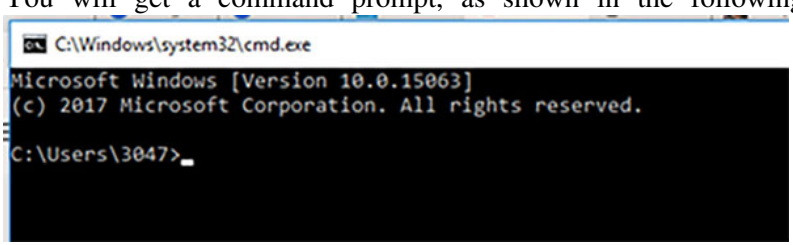


Figure 5.13: *Screenshot of Point 5 (Command prompt)*

6. Now, write **"cd Desktop"** at the command prompt. You get **"C:\Users\3047\Desktop>**

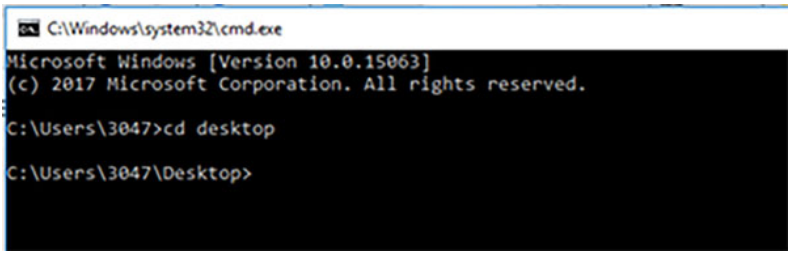


Figure 5.14: *Screenshot of the output of Point 6 (DeskTop)*

7. At the command prompt, write the command `C:\Users\3047\Desktop>python get-pip.py` (ensure that your internet connection is on). The Pip package will start extracting and will take some time. Wait for almost a minute, and you will get a command prompt indicating that the pip is installed successfully.

Is it already installed on your system, it will show the appropriate message as shown on the screen as follows:

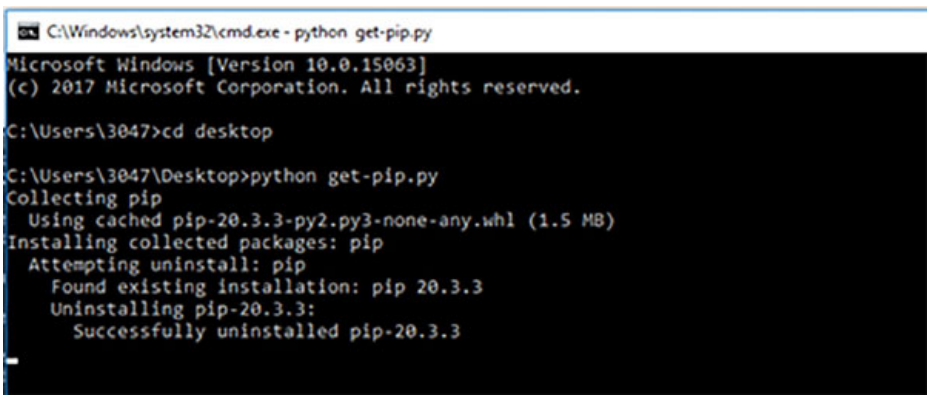


Figure 5.15: *Screenshot of of output of Point 7 (pip)*

Python pip is now installed on your system.

Installing Beautiful Soup

Now we install the Beautiful Soup library. We will be using the library a lot in this book. Just follow the following steps to get it installed on your machine.

1. Write `pip install beautifulsoup4` at the command prompt. You will see the installation process will start and within a few seconds, you will get a message "`Successfully installed Beautiful Soup 4.9.3`" message that confirms that the library is installed successfully. In case it is already installed, you will get a message as shown in the following figure:

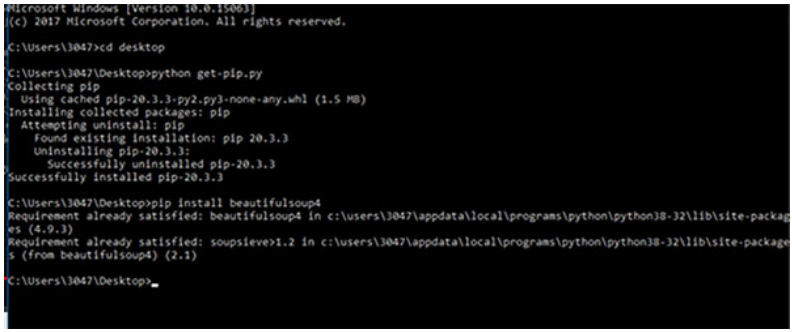


Figure 5.16: *Screenshot of of output of Point 1 (installation of BeautifulSoup)*

2. To confirm that BeautifulSoup is installed, open the Python prompt and give the following command:

```
>>> from bs4 import BeautifulSoup
```

3. If there is no error (traceback), it means you are ready to use the library.

Conclusion

This chapter discussed the concepts of Web scraping, its uses and how it works. It also discussed about various components in a Web scraper, Python modules, and how these modules are useful in scraping. This chapter also discusses about legality of Web scraping and the cautions during Web scraping process. It was also discussed as to importance of data extraction and preprocessing before it can be used for meaningful analysis/use, followed by learning the core concepts of scraping a text, image, and videos with examples. This chapter also talked about scraping dynamic websites, later a discussion on use of CAPTCHA, its types and ways to handle CAPTCHA during scraping was also discussed. At the end a case study showing the implementation is also discussed.

In the upcoming chapter, we will discuss the concepts of Web Opinion Mining at different levels and its applications in various day-to-day works. We will also learn about the data sources and various preprocessing steps for text processing and cleaning of data, such as tokenization, feature extraction, and so on.

Points to remember

- Web scraping, often referred to as Web harvesting or Web extraction, is a technique that is used to extract useful information from the World Wide Web (WWW).
- The information extracted is saved to a file or a database for later use in its analysis. Web scraping is a technological solution to extract data from websites in a quick, efficient, and automated manner.
- Web scraping helps to extract relevant data from unstructured websites and transforms and stores it in a structured format like a csv file or database,

which can then be used to analyze for making effective decisions.

- But in general, it will follow the following process:
- The Web scraper will be given one more URL (from where the data is to be retrieved).
- The scraper program sends the GET request to the server and returns a response in the form of Web content.
- Due to the dynamic nature of the Web and its contents, Web scraping is also full of challenges like ethical changes, rapidly changing technology while website creation, varying site structure, and so on.
- A Web scraper may be developed using any of the programming languages such as Java, Python, Ruby, or JavaScript in the framework Node.js. This book uses Python libraries to implement the scraping.
- Another approach is to use of the framework, which takes care of each step in the process. Python provides a framework called scrapy, that provides functions for each step in the process of scraping. Most of the website use robots.txt file to specify what contents scrapers can scrape and what they should not touch.
- Python library called "`b u i l t w` " and "`B e a u t i f u l S o u p` " is used for handling text, image, and videos.
- Captcha is a visual interface to stop automated computer programs, also known as bots and spiders, from gaining access to the website.
- **CAPTCHA**, which is an acronym of "**Completely Automated Public Turing test to tell Computers and Humans apart**".
- Based on the level of complexity, different types of CAPTCHA are being used. Some of the most common ones are listed as Character Recognition, Audio Recognition, Math Problems, Image Recognition, Social Media Logins.

Multiple choice questions

1. What protocol can be used to retrieve Web pages using Python?

    a. urllib

    b. bs4

    c. HTTP

    d. GET

2. What provides two-way communication between two different programs in a network?

    a. Socket

    b. Port

c. HTTP

d. Protocol

3. What is a Python library that can be used to send and receive data over HTTP?

   a. HTTP

   b. urllib

   c. Port

   d. Header

4. What does the following block of code do?

```
import urllib.request
fhan=durllib.request.urlopen('http
data.pr4e.org/romeo.txt')
for line in fhand:
 print(line.decode().strip())
```

   a. It creates a file named 'romeo.txt' in 'data.pr4e.org'

   b. It finds the URLs linked to 'data.pr4e.org' and prints it.

   c. It opens a file named 'http://data.pr4e.org/romeo.txt' in local storage

   d. It prints the contents of 'romeo.txt' after retrieving it from 'data.pr4e.org'

5. What does the following block of code do?

```
import urllib.re,qurelsltib.p aurrslel ib.err
img= urllib.request.urlopen('http:
cover3.jpg').read()
fhan=dopen('cover3'.wjbp'g)',
fhand.write(img)
fhand.close()
```

   a. It retrieves 'cover3.jpg' and saves it to your computer.

   b. It displays the image 'cover3.jpg'.

   c. It retrieves the URL to download 'cover3.jpg'

   d. None of these

6. What does the following block of code do?

```
url= "https://www.nytimes.com"
htm± urllib.request.urlopen(url,
context=ctx).read()
soup BeautifulSoup(html,'html.parser')
```

   a. Retrieves and displays the Web page

   b. Parses the HTML content of the "https://www.nytimes.com" Web page.

c. Downloads the Web page

d. Both a and b

7. What does the following block of code print?

```
url = "https://www.nytimes.com/"
html = urllib.request.urlopen(url).
soup = BeautifulSoup(html, 'html.parser')
tags = soup('img')
for tag in tags:
  print(tag.get('src',
```

a. Retrieves and displays the Web page

b. Downloads the Web page

c. Prints the images from 'www.nytimes.com'

d. Prints all the 'img' sources under 'src' from 'www.nytimes.com'

8. Acronym of CAPTCHA is _____.

a. Computer-based Automated Public Turing test to tell Computers and Humans apart

b. Completely Automated Public Turing test to tell Computers and Humans apart

c. Computer-based Automated Programme Testing test to tell Computers and Humans apart

d. None of the above

9. _____ is the Python framework used for Web scraping.

a. Beautiful Soup

b. urllib

c. scrapy

d. builtwith

10. _____ file of a website that contains instructions for a Web scraper to specify what to touch and what not to touch.

a. Node.js

b. Scrapy

c. Robots.txt

d. None of the above

Answers

Questions

1. Define Web scraping? Explain with a diagram the working of a scraper?
2. Write about the uses of Web scraping?
3. Discuss the various components of a Web scraping?
4. There is a lot of debate regarding the legality of Web scraping. Write your views about this debate.
5. What do you understand by CAPTCHA? Discuss about different type of CAPTCHA. How to deal with websites with CAPTCHA?
6. Discuss the Python modules that are used for Web scraping.

Key terms

- Web scraping
- Web harvesting
- Web scraper
- Preprocessing
- Dynamic website
- CAPTCHA

CHAPTER 6

Web Opinion Mining

Introduction

The latest field of study, Web opinion mining, is a data mining technique for extracting opinions from the Web. It helps in analyzing the data and opinions from various Web sources and provides meaningful insights to users for better decision-making. With the proliferation of Web 2.0, people contribute more time toward sharing their thoughts on various social media platforms such as Facebook, Twitter, and so on, which leads to rapid growth in the field of Sentiment Analysis. *NLP is a technique which provides useful models for determining the emotions and sentiments from communications. To build NLP models,* ***Natural Language Toolkit*** *(**NLTK**) is one of the prominent libraries that helps build these models and makes Sentiment Analysis the uppermost solution.* In this chapter, you will get to learn about the concepts of opinion mining and various NLP techniques for processing the data. We will also discuss a case study on Twitter data for sentiment analysis by providing in-depth knowledge about various libraries of NLP.

Structure

In this chapter, we will discuss the following topics:

- Concepts of opinion mining
- Collection of review
- Working with data
- Data preprocessing and tokenization
- Part of speech tagging
- Feature extraction
- Case study on YouTube comments/Twitter

Objectives

In this chapter, we will learn about opinion mining, data cleaning and various important processing steps for cleaning the data using Natural processing techniques such as tokenization and part-of-speech tagging. A case study is also discussed for sentiment analysis on Twitter data.

Concepts of opinion mining

Unlike factual information, opinions or sentiments are subjective in nature. Due to a large number of opinions and related sentiments on the Web related to any product, a summary of opinions becomes important (to understand the overall sentiment associated with it).

For example, if we have a statement:

Example 1: "My Nikon camera has a very good picture quality. I am very much satisfied with it, but my wife says it is very heavy to carry"

Both positive and negative sentiments are associated with this sentence, "good quality" and "very much satisfied" display positive emotion/emotion and "very heavy" show negative sentiment/emotion.

"The study of human emotions and its analysis through computer modeling techniques is termed as Sentiment Analysis or Opinion Mining". It is a computational study of the sentiments and attitudes expressed by the user or speaker related to any topic, service, product, and so on. With the emergence of Internet 2.0, social media platforms are gaining much importance nowadays for sharing thoughts and views. For example, if a person wants to buy a mobile phone, he will consult his friends and relatives about the various features of the phone, such as its camera quality, battery life, calling quality, and so on. From the reviews he receives from his friends, he will be able to make a better decision for buying the phone. Thus, for better decision-making, the advice of others is important so that spending money on useless products or services can be avoided. But, with the growing era of the internet, reviews related to any other product are not limited to asking in person but can be extracted from various social media platforms. This results in a lot of opinionated data on various sites such as Facebook, Twitter, and so on. The amount of data that is recorded in the form of digital text needs some analysis to extract some meaningful data and set valuable information for any business, customers, or reviewers to improve their services, as shown in the following figure:

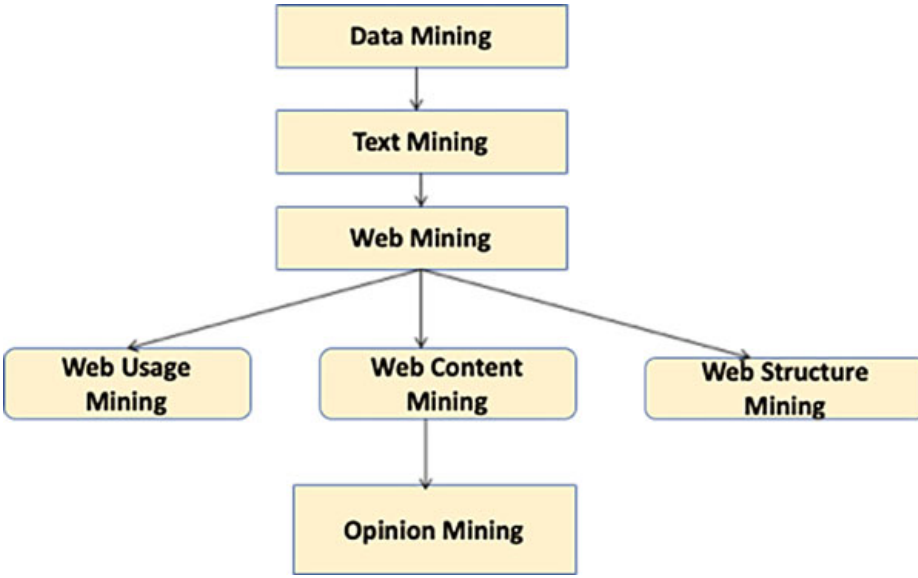


Figure 6.1: *Hierarchy of opinion mining*

"Data Mining" is broadly classified into "Text Mining" and "Web Mining". Opinion mining is further classified by Web content mining. As the name suggests, in Web content mining, various contents from the Web are mined for meaningful information for better decision-making. Opinions and their related components, such as emotions, sentiments, and attitudes, are the subjects to study and analyze for the task of Sentiment Analysis. Various **Natural Language Processing** (**NLP**) techniques are used to discover, retrieve and distill the information from huge databanks available on the internet. *Figure 6.1* shows the hierarchy of opinion mining.

Texts or reviews available on the internet can be classified as opinions and facts. By Facts, we mean the actual thing that exists and can be proven. A fact is an objective information about any entity. However, opinions are subjective information, and any individual can agree or disagree with one's opinion. Opinion mining is a classification task that classifies the text into two important categories:

- Binary Classification—(Positive and Negative)
- Ternary/Multiclass Classification (Positive, Negative, and Neutral)

There are various types of Sentiment Analysis polarity classification (binary and ternary), emotion detection (happy, sad, and angry), topic-wise sentiment analysis (product reviews and online shopping), and so on. Positive and negative emotions are connected with one's emotions of being happy, sad, anger, and so on, leading to forming of positive or negative opinions about the product.

The various characteristics associated with Sentiment Analysis are as follows:

- **Target Holder**: The one who expresses his/her views like a reviewer or user

- **Target**: The one for which you are expressing your views
- **Polarity**: The classification of text, whether positive or negative

This can be easily understood by the example shown in *figure 6.2*:

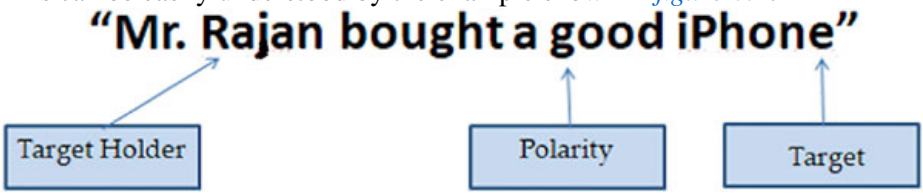


Figure 6.2: *Characteristics of Sentiment Analysis*

Target Holder: Mr. X (He is the holder who expresses his opinion about a target that is iPhone bought by Mr. Rajan).

Target: iPhone (For which the opinion is expressed)

Polarity/Sentiment: Good (Sentiment expressed)

Definition

Opinion mining was first introduced by *Nasukawa* and *YI* in the year 2003. Sentiment Analysis refers to the task of systematically analyzing subjective information (with respect to opinion/sentiment) with the use of natural language processing, text analysis, and computational linguistics.

NLTK for sentiment analysis

The Natural Language toolkit, also known as NLTK, is the API for natural language processing. NLTK works with the Python programming language. It was introduced by Steven Bird and Edward Loper for statistical natural language processing of the English language. It is a free, open-source library available for Windows, Linux, and so on. It can be installed from "nltk.org". NLTK is the most popular library to deal with problems related to linguistic data.

Every natural language processing task requires preprocessing of data for feeding data into various machine learning algorithms. The data extracted from various data sources available on the internet is usually unstructured. Here, by unstructured data, we mean it is not in a uniform format, and to make it structured, the NLTK toolkit is used to preprocess the data, which results the data in a structured format. It helps in removing noisy data, stop words (a, an, the, and so on), punctuation marks and converting text into lowercase, tokenization, and so on, and thus, resulting in the data in the structured format.

Opinion Mining/Sentiment Analysis at different levels

Sentiment Analysis, also known as Web Opinion Mining, is categorized into different levels depending upon the sentiment extracted from the piece of text or reviews. It can be broadly classified into word, sentence, document, and feature

level, as shown here in the following figure:

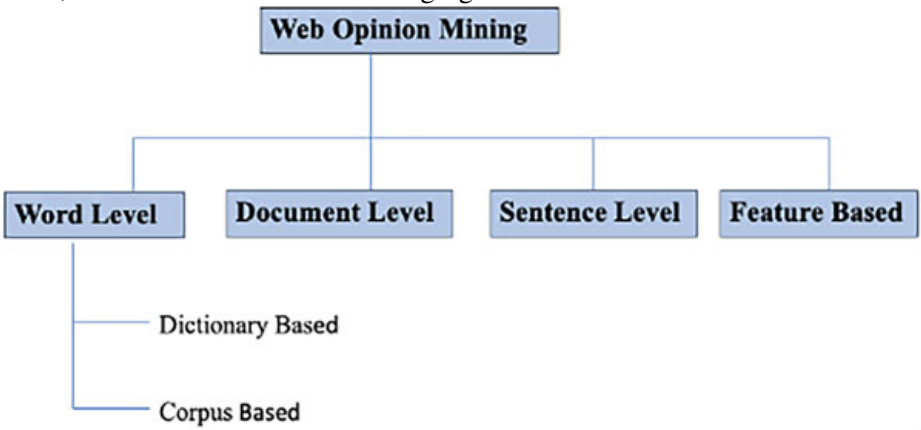


Figure 6.3: *Levels of Sentiment Analysis*

Word level

Sentiment Analysis of the word level is used to find semantic Phrase-level Orientation. Two methods of annotating sentiment automatically at the word level are dictionaries and corpus. Dictionaries contain predefined words expressing the sentiments as positive, negative, and neutral. SenticNet and SentiWordNet are some of the dictionaries available for the dictionary-based approach. This approach is used to find the context of a sentence in sentiment analysis.

Document level

In document-level opinion mining, the whole document is classified into positive or negative opinion aspects. In this, a review is extracted on a single object.

Example 2: The Prime Minister of India, Mr. Narendra Modi, is a nice person. He is a visionary leader and a good speaker. He initiated the "SWACH BHARAT" campaign, but due to less awareness among people, this mission was unsuccessful.

From the preceding example, we can classify our documents as follows:

Positive Document:

- Mr. Narendra Modi is a nice person.
- He is a good speaker.

Negative Document:

- Failure of the "Swachh Bharat" campaign.

Sentence level

In sentence level Opinion Mining, the emphasis is on sentences. In this polarity of sentences, it is reviewed whether the sentence is positive, negative, or neutral. Neutral means no opinion is expressed. This level of analysis is closely related to Subjectivity Classification (Weibe, Bruce, and O'Hara 1999), which distinguishes a sentence (called an objective sentence) that expresses factual information from a sentence (called a subjective sentence) that expresses subjective views and opinions.

Example 3: "I bought a Motorola phone two weeks ago. Everything was good initially. The voice was clear, and the battery life was good, although it is a bit bulky. But, it stopped working yesterday!!"

In the preceding example, we focus on different sentences and review the polarity of sentences, whether it is positive, neutral, or negative.

Feature-based

The previous section described two levels of opinion mining level. It did not describe the exact opinion of people, that is, whether the people liked or disliked a particular feature of any product. Instead of looking at language constructs (documents, paragraphs, sentences, clauses, or phrases), aspect level directly looks at attributes of entities.

Example 4: If a user post reviews about a feature like the battery life of an object (Mobile), he is commenting about the attribute of a phone (battery). So, if a customer wants to buy a mobile with a good quality battery, he can decide easily by this feature whether to purchase this mobile or not.

Collection of review

For the task of Sentiment Analysis, the collection of data/ reviews is an important task. There are various data sources available for the extraction of reviews from social networking sites and many other resources. We will discuss some widely used data sources for opinion mining.

Data sources for opinion mining

In the era of Web 2.0, there is a huge amount of data available for the analysis of sentiments, and the data volume is growing very fast. There is a huge data bank of texts, videos, images, and audios. So, here you will get to know about some important data sources available for extracting opinions.

Blogs

A blog is a website where people share their thoughts or opinions regularly. A blog is a personal diary where people write regularly and share their feelings about any

topic. A person who writes content on the blog is known as a "blogger". Content in a blog could consist of images, media files (audio and video), images, and so on. Various blogs discuss views on different categories, such as restaurants, movies, gadgets, tourism, fashion, music, and so on. Thus, from blogs, we get a lot of data for analysis.

Review sites

Review sites are a great source of data. These sites are used to share their reviews about any product, clothing brand, and so on. On review sites, you can get a large amount of data, which can be positive or negative and results in a better decision-making process for an individual to rely upon. Various e-commerce sites such as Amazon, Flipkart, Myntra, and so on provide reviews about products that you can view and decide accordingly for your purchase. These review sites are a great medium for various brands to improve their product by knowing customer's points of view and what they expect in terms of quality and services.

Forums

A forum is also known as a message board or internet forum, where people post their messages regarding any topic of discussion. People can start their discussion with a thread, and it can remain there for years and some conversations can also lead to heated arguments. For this to control, there are certain rules, which are created by the moderator of the forum. Thus, the forum is again a good data source for getting a large amount of data.

Social networking sites

The most popular data sources nowadays are social networking sites. Facebook, Twitter, LinkedIn, and Pinterest are some of the most popular social sites, which are used by most of us to share our thoughts and opinions. These sites help in building social connections with people having similar interests. Twitter is a well-liked social platform among all individuals, and it has almost 300 million users. It is an excellent platform for politicians to share their thoughts and even for businesses to reach to their valued customers.

Working with data

Natural language processing is mainly concerned with the interaction of computers with natural language. Artificial Intelligence is an emerging field, and it provides good insight into making computers understand human speech. This ability to understand human language or views is accomplished through the task of Machine learning. Now, we will explore the various operations which are performed on the text through NLP.

The string data type is used for text representation in any programming languages

such as C, C++, Java, and Python. There are various operations performed on strings which we will discuss now with the help of various tasks using NLP.

Pre-processing of data

Cleaning of raw data is required for further processing of the Sentiment Analysis task. There are a few steps of preprocessing done that result in cleaned and organized data that can upgrade the results of data mining problems. It is a very crucial step in the mining process and before feeding the data to any algorithm. Machine learning models require structured and organized data for training any model. Various classification algorithms can apply to this organized data for text classification. The data available on various social media platforms is unstructured, and by applying various preprocessing steps, the data is converted into clean data. Here, you will see various preprocessing steps required for cleaning the data.

Tokenization

Tokenization is one of the most important tasks when we work with textual data. It is the method for splitting sentences, paragraphs, documents, and phrases into small units. These smaller units are called tokens. For performing the task of tokenization, the NLTK is a widely used Python library. Let us see the process of tokenization at different stages.

Sentence tokenization

Sentence tokenization is used to split the text into individual sentences. **sent_tokenize** function is used from NLTK to perform this task of tokenization of text into sentences. The arguments for this function will be text which we want to tokenize.

Example 5:

```
1.import nltk
2.nltk.download('punkt')
3.nltk.download('averaged_perceptro
4.from nltk.tokenize import sent_tokenize
5.txt= "I love watch images and my friends love
eating pop corns and snacks while watching Movie."
6.tokenized= sent_tokenize(txt)
7.print(tokenized)
```

Output:

```
['I love watch images! I and my friends love
eating pop corns and snacks while watching Movie.']
```

Note:

1. Lines 2 and 3 may be skipped if these libraries are already downloaded on your system
2. "punkt" is used for tokenizing sentences, and "averaged_perceptron_tagger" is used for tagging words with their parts of speech (POS).

In this example, a document is tokenized in two different sentences *"I love watching movies." and "I and my friends love eating popcorn and snacks while watching Movie."* Thus, the whole text is split into different sentences by the **sent_tokenize** module. You might have a question in mind as to what is the need for sentence tokenization when word tokenization is available. Suppose you have to count the average words per sentence; then you need to split the text into sentences.

Word tokenization

By performing word tokenization, we split the sentences into separate words. To perform word tokenization, we use **word_tokenize** function, one of the TreebankWordTokenizer instance of NLTK. The argument for this function will be tokenized sentences, which are done with the help of **sent_tokenize** function. This is shown in the following code snippet:

Example 6:

```
1.import nltk
2.from nltk.tokenize import word_tokenize,
sent_tokenize
3.txt= "I love watching movies. I and my friends
love eating popcorn and snacks while watching Movie."
4.tokenized = sent_tokenize(txt)
5.for i in tokenized:
6.    wordsList = nltk.word_tokenize(i)
7.    print(wordsList)
```

Output:

```
['I', 'love', 'watching', 'movies', '.']
['I', 'and', 'my', 'friends', 'love', 'eating',
'popcorn', 'and', 'snacks', 'while', 'watching',
'Movie', '.']
```

In the preceding example, the words of a sentence are tokenized into individual tokens. First, the sentence is tokenized by using the **sent_tokenize** function and then the tokenized sentence is further tokenized into individual tokens by using the **word_tokenize** function.

Note: Tokenization is important as it will be easy to interpret the meaning of the

text by splitting it into tokens.

Part of Speech tagging

Part of Speech tagging is one of the crucial steps in NLP. With the help of POS tagging, a particular part-of-speech is tagged to individual words used in a sentence. These tags define whether the word is a noun, adjective, verb, adverb, and so on.

These taggers help in identifying the category of tokens used in sentences which can either be nouns, verbs, and so on. `nltk.tag` package is inherited from the TaggerIbase class for generating taggers for various words.

The list of tags available for Part of Speech is as follows:

CHAPTER 7

Web Structure Mining

Introduction

The structure of a Web typically consists of Web pages and hyperlinks connecting them. Structure mining is used to generate the structural details of the website. It discovers the link structure of the hyperlinks within the Web. Web structure mining helps us to identify similar websites, classify the website, to establish the relationship between websites, or discover websites. This chapter begins with a discussion on the concept of Web structure mining and its types, followed by other related topics such as Web graph mining, Web Information extraction, Web search and hyperlinks, **Hyperlink Induced Topic Search** (**HITS**) search, partitioning algorithms used for mining, and a case study demonstrating the concepts studied in the chapter.

Structure

In this chapter, we will cover the following topics:

- Introduction
- Concept of Web-structured mining
- Types of Web-structured mining
- Web graph mining
- Web information extraction
- Deep Web mining
- Web search and hyperlinks
- Hyperlink-induced topic search (HITS)
- Partitioning algorithm
- Implementation in Python

    Set UP graph

    Iteration

    Return

Objectives

After studying this chapter, you will be able to understand the concept and type of Web mining. You will learn how the Web can be represented as Web graph, Web

graph mining, Web information extraction, and deep learning. This chapter will help you in applying Web search and Web search algorithms. We will also work on a case study on community detection

Introduction to Web structure mining

In *Chapter 1, Introduction*, we discussed three types of mining, Web content mining, Web structure mining, and Web usage mining. Till now, we have been discussing web content mining that talks about scraping various types of content, such as text, images, video, captcha, and so on. In this chapter, we begin with a discussion on Web structure mining.

Due to the increase in the amount of data available online, the World Wide Web has become one of the most valuable resources for information retrieval and knowledge discoveries. The internet is flooded with many Web applications and has also become a very convenient place of a lookout for any kind of information. All this is available in the form of Web pages that are characterized by Weblink and Hyperlink:

- **Weblink:** The link or reference of the Web page through which a page is opened, irrespective of on which Web server it is stored.
- **Hyperlink:** The link within the Web page that can be used to connect to any other location of the Web page or any other Web page of some other website. A hyperlink could be understood as the connection point between two pages and can be a text or an image. Text or image that acts as a hyperlinked is clicked to retrieve the document, which is connected through the link.

Most of the Web pages are designed using HTML code. HTML code is in semi-structured formal; thus, the data of Web pages are in semi-structured format.

- There are links between information on the pages of a site and between pages from different sites.
- The same information or similar versions of it can appear on multiple pages.
- Information can be found on the surface (in pages that are accessed via browsers) or deep (in the database, there are queries through different interfaces).
- Web pages are of dynamic nature; thus, information is also dynamic. Constant monitoring of changes in the information is an important issue and becomes challenging.
- Above all, the internet has become a virtual company. In addition to information and services contained, the internet offers the possibility of interaction between people, thus, contributing to the creation and development of new communities.

Web structure mining may be defined as a process in which structure information is discovered from the Web. In this chapter, we will discuss Web structure mining

and other topics associated with it.

Concepts of Web structure mining

Web structure mining is the process to extract patterns from hyperlinks on the Web. It is also called link mining. We can understand the users' navigational patterns using Web structure mining which helps to reveal the users' interests.

The structure of a website is usually based on how the designer envisions the site will be used. However, once the website is put into use, the designer's theoretical approach may turn out to be not so practical. It is only the actual use of the website that will give the designer clues about how the users navigate through the site and in what content they are most interested. For example, if the users are clustering toward a particular type of content, then the designer could think about establishing the site as an authority on that type of content by providing more information on the topic. Conversely, if a type of content is not generating much attention, it may be because the users are missing the information. The designer may look into changing its location on the website. The users' navigational patterns help reveal the users' interests, but they can also be used to adjust the hyperlink structure of the website for optimal navigation

Web structure mining is discovering structure information from the Web. For this purpose, the Web structure is represented in the form of a Web graph in which Web pages as represented as nodes and hyperlinks as edges of connecting related pages. Structure mining is used to get a structured summary of a particular website. It looks for the relationship between Web pages linked by information or direct link connection. Web structure mining can be very useful to determine the connection between two commercial websites.

Web structure mining

Web structure mining has become one of the core techniques of Web mining that deals with hyperlink structure. It helps in identifying the relationship between linked Web pages of the website. Various algorithmic techniques can be employed to discover data from the Web. Structure mining analyzes hyperlinks of the website to collect informative data. Probably, the most useful example of exploiting link structure is the use of links to improve information retrieval results. This link information is intra-page, which is done at the document level, and the hyperlink level is inter-page mining.

The purpose of Structure Mining is to produce the structural summary of websites and similar Web pages. This type of mining is applied at the level of document and hyperlink level. Web Structure Mining plays a very important role in the mining process.

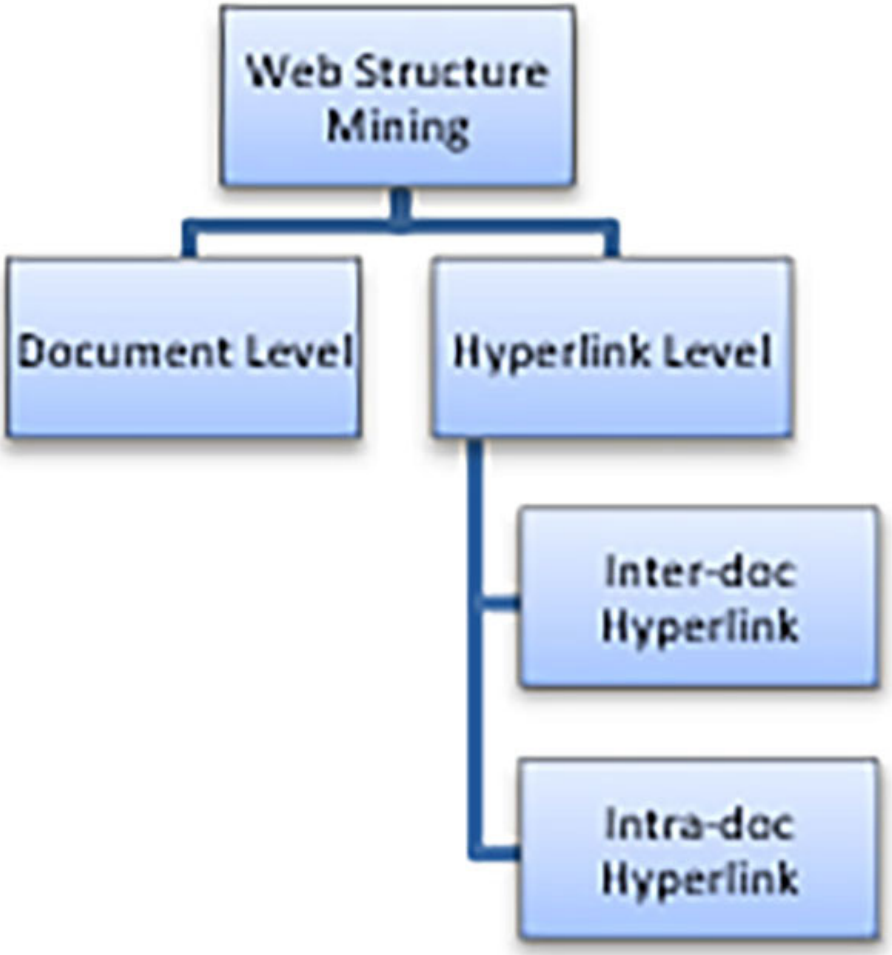


Figure 7.1: *Types of Web structure mining*

In document-level mining, we try to analyze the structure of the document (Web page/document), whereas hyperlink-level structure mining discusses about Inter-doc hyperlinks (in which we try to find the relationship between the links within the Web document) and Intra-doc hyperlinks (the purpose is to find the relationship between various links visited on different Web pages). In this chapter, we are concerned with hyperlink mining, which may be used for discoverability.

Structure analysis is also called link mining, and various tasks associated with Web structure mining are given as follows:

- **Link-based Classification:** In Link-based classification, Web pages are classified using the relations or links present amongst them.
- **Link-based Cluster analysis:** In clusters-based analysis, the data/Web pages is/are grouped into similar and dissimilar objects/ groups termed as clusters.
- **Link Type:** Link type analysis is involved with the analysis of the different

types of links, that is, the internal links (links within the same domain), external links (links in an outside domain), and backlinks (the links that bring back to the domain from where you left to visit the external domain).

- **Backlinks:** Backlinks are links from one page on one website to another. If someone links to your site, then you have a backlink from them. If you link to another website, then they have a backlink from you. Search engines like Google consider backlinks as votes of confidence. It is considered, that the more votes your Web pages have, the more likely they are to rank for relevant search queries.
- **Link Strength:** Link Strength is a metric to calculate the overall strength of backlinks. The Link Strength ranges on a scale from 0 to 100. The higher the score, the more powerful the backlink is.
- **Link Cardinality:** Cardinality indicates the number of occurrences (one or many) that one entity has relative to another. The main task in finding link cardinality is to predict the number of links between objects, Page categorization, finding related pages, finding duplicated websites, and looking for the similarity between them.

Probably the most famous example of exploiting link structure is the use of links to improve information retrieval results. Both the well-known link analysis algorithms' page rank and hubs and authority scores are based on the link structure of the Web. These algorithms work using in-links and out-links of the Web pages to evaluate the importance or relevance of a Web page.

Many analysis-based algorithms have been proposed. We will discuss two important such algorithms, PageRank, and HITS, in the coming section of this chapter.

Web graph mining

Several Web pages are connected with other Web pages through something that is known as a hyperlink. We have also discussed that the world wide Web is a collection of billions of Web pages, and most of them are connected through hyperlinks to each other. These Web pages contain lots of relevant and related information. Let us say we want to search for some information about "Web Mining". There will be thousands of Web pages that contain information about this term. Now, how do we decide which page will give the most appropriate information? Web graph-related algorithms will help us to do so. To understand this, we need to have an understanding of Web graphs.

The Web is represented in the form of a graph, as in mathematics, and relationships can be expressed in terms of a graph. A graph represents the vertices (or nodes) connected by lines (edges) or curves (see *figure 7.2*). We can also say that a graph is a collection of nodes and edges that represents relationships between nodes and edges.

The graph edges may have a weight that indicates the strength of the connection between the objects:

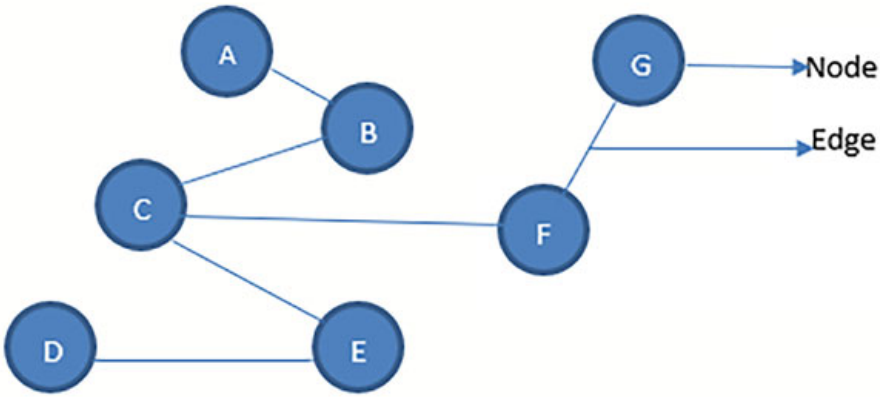


Figure 7.2: *Graph with Nodes and Edges*

Some of the examples where graphs may be of use are to model friendship in social network analysis (discussed in detail in the upcoming chapter), to represent the relationship between various Web pages in the world wide Web (Web page linking), or to represent flights between various airports (where the airport is a node and the flight between airports are edges), and so on.

Based on the type of relationship, the graphs may be Undirected or Directed. In an undirected graph (*figure 7.1*), the edges do not have a direction and represent a two-way (mutual) relationship (the edges can traverse in both directions), whereas, in a directed graph (*figure 7.2*), the edges have a direction that represents a one-way relationship (each edge can be traversed in a single direction).

Figure 7.2 represents an undirected graph with nodes (vertices) A, B, C, D, E, F, and G, and the edges AB, BC, CE, ED, CF, and FG. We can observe that the edges do not have any direction; thus, we may write an edge as AB or BA, and the order will not matter.

Figure 7.3 represents a directed graph with nodes A, B, C, D, E, F, and G, and the edges AB, BC, CE, ED, CF, and FG. We can observe that the edges have a specific direction, which means the relationship is not mutual, and while writing the edges, AB is not the same as BA, as AB represents the relation from “A” to “B”, whereas BA will mean vice versa.

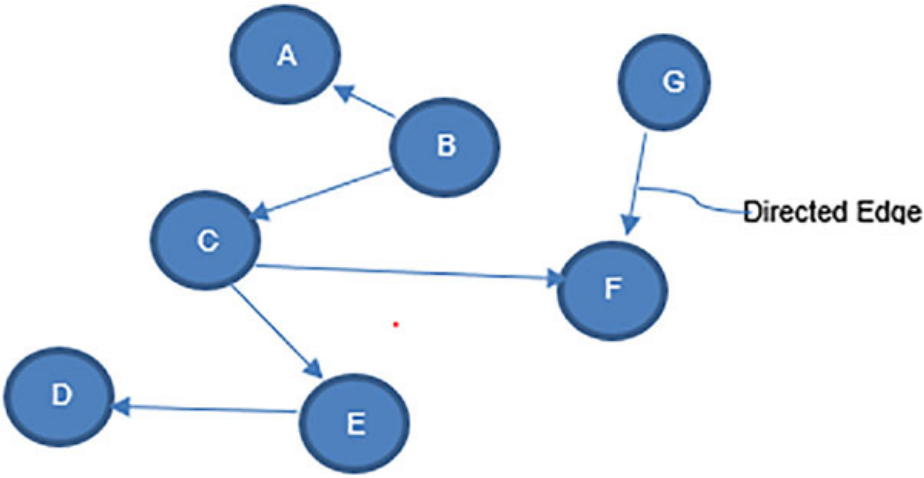


Figure 7.3: *Directed graph with nodes and edges*

The Web graph describes the world wide Web as the directed graph between various pages on the Web. It consists of several nodes (the Web page) and directed edges (the links) connecting each Web page in the Web (see *figure 7.4*).

Each edge represents a link either coming TO a website or leaving FROM a website. In the case of our example, as follows, the link takes us out from a Web page(node) A to a Web page(node) B, Node F links to node B, and so on.

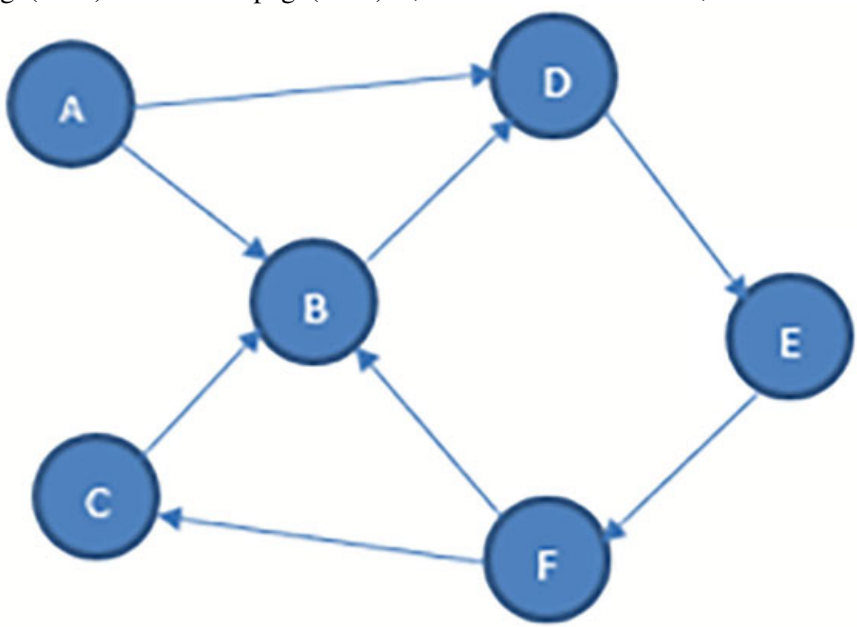


Figure 7.4: *Web as a directed graph*

Figure 7.4 can be seen as a pictorial representation of the worldwide Web where nodes can be understood as Web pages, and the edges represent the Web links

between the given pages. Hyperlinks that move into a page as known as in-links, and those moving out of a page are known as out-links. As per studies, on average, the number of in-links to a page (in-degree of the Web page) is roughly between 8 and 15. Similar to in-degree, the number of links out of the Web page is defined as the out-degree of a Web page.

In this example, we consider six Web pages labeled as A-F. Page B has in-degree 3 and out-degree 1. This graph is not strongly connected as there is no path from any of pages B and F to page A. As per the research, the average node in a large website has roughly seven hyperlinks (directed edges) to other pages, which makes several million hyperlinks.

Web graphs may be used for various applications as follows:

- Computing the PageRank of Web pages
- Computing personalized PageRank
- Detecting Web pages of similar topics through graph theory properties like citations
- Identifying hubs and authorities for the HITS algorithm
- Improved Web search
- An improved topic classification algorithm
- Enumerating emergent cyber communities

Set of tools and techniques used to analyze the properties of real-world Web graphs (discussed in detail in *Chapter 8, Social Network Analysis*), predict how the structure and properties of a given Web graph might affect some applications, and develop models that can generate realistic graphs that match the patterns found in real-world Web graphs of interest is known as Web graph mining.

Web information extraction

"Information Extraction (**IE**) is the process used for automatic extraction of a structured information from structured or semi-structured text documents using various tools and techniques." In the mid-1990s, when the amount of unstructured content was continuously growing on the Web, one of the earliest methods of information extraction was screen scraping, which was used to compare the prices and other details of any product from various online websites for analysis purposes (for example to find the price, of a book with the same ISBN on various Web sites). Recently, more sophisticated techniques have been used on the Web to improve search quality. Amongst various IE applications, the important ones are entity extraction and relationship extraction.

Application of IE techniques to process the vast amount of unstructured content on the Web is termed as Web information mining. Web information extraction may be used to extract reviews, opinions, and sentiments also, along with the named entity and relationship extraction.

Entity Extraction

It refers to the identification of mentions of the named entities (such as the name of a person, location, organization, and so on) from the unstructured content on the Web. These are usually the subjects or objects in the sentence. For example, let us consider the sentence:

"The Prime minister will meet the Chief Minister of the state in the capital city"

In this sentence, various entities that represent a person or place are as follows:

The Prime minister

The Chief minister

The capital city

Named entities are entities that are represented by a proper noun. For example, if we have the sentence as follows:

"Dr. Manmohan Singh will meet the economist, Mr. Abhijeet Banerjee, in New Delhi"

In this, the entities are Dr. Manmohan Singh, economist, Mr. Abhijeet Banerjee, and New Delhi. But the named entities are only Dr. Manmohan Singh, Mr. Abhijeet Banerjee, and New Delhi, as they are the proper nouns as part of the speech tag.

For example, for recognizing names as named entities, due to different conventions, short hands, and formats used in writing names all over the world, several rules are required to capture well-formed names from the text with consistent and complete syntax.

Person names: "Abdul Kalam", "Rajiv Gandhi", or "Dr. Albert Einstein" or "Professor Dr. Homi Jehangir Bhabha"

Relationship Extraction

Relationship extraction is used to create structured knowledge bases from unstructured text or Web pages. This type of extraction deals with associating pairs of named entities. For instance, *figure 7.5* shows a snapshot of the Wikipedia page of IIT Delhi shows a piece of tabular information on the right-hand side as shown in the figure:

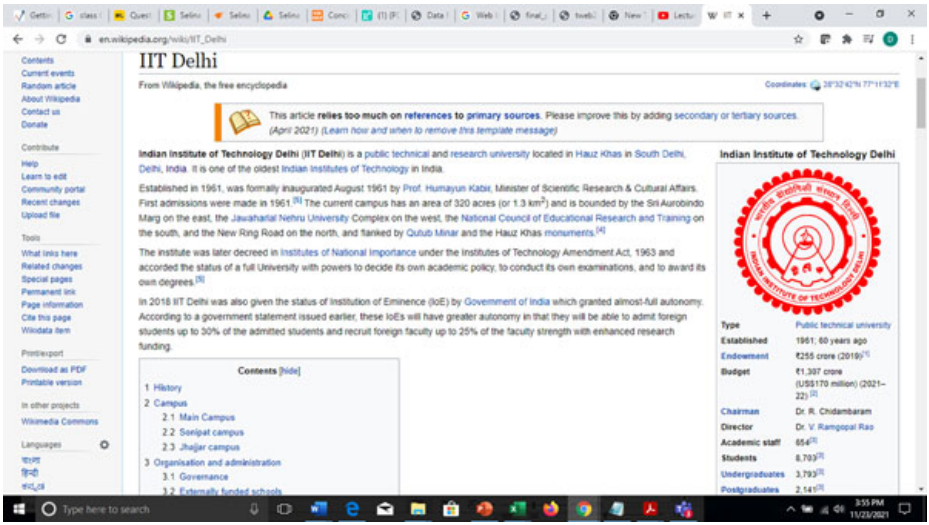


Figure 7.5: *Various relationships from the given Wikipedia*

Figure 7.6 shows various relationships from the given Wikipedia page on the right-hand side as follows:

Established: 1961 (represents establishment year)

Chairman: Dr. R. Chidambaram

Director: Dr. V. Ramgopal Rao

Academic Staff: 654

Students: 8703

and so on

Indian Institute of Technology Delhi

Type	Public technical university
Established	1961; 60 years ago
Endowment	₹255 crore (2019)[1]
Budget	₹1,307 crore (US$170 million) (2021–22) [2]
Chairman	Dr. R. Chidambaram
Director	Dr. V. Ramgopal Rao
Academic staff	654[3]
Students	8,703[3]
Undergraduates	3,793[3]
Postgraduates	2,141[3]

Figure 7.6: *Extracted relationships from the given Wikipedia*

Reviews, Opinion, or Sentiment analysis

Social media has become ubiquitous (present everywhere) in the lives of consumers. Thus, there is a constant interest in the techniques to reliably extract reviews, opinions, and sentiments about various products and services from the content present on various online Web pages. This information can be used for formulating the right business strategies. For example, a laptop company would be benefitted from the sentiments or opinions related to their product, as perceived by the customers (see the previous chapter).

Due to the tremendous increase in the size of the Web, there is growing interest amongst various communities in exploring the use of Web information extraction tools and techniques for the extraction of meaningful information/data. IE techniques are being used in varied areas such as Web search, online communities, sentiment analysis, Web analytics, and so on.

Extraction of reviews, opinion and analysis of sentiment associated with them has been discussed in details in *Chapter 6, Web Opinion Mining*.

Deep Web mining

Search engines such as Google, Bing, and Yahoo search and index websites because of links. Search results are ranked on the basis of these links on the basis of relevancy, inbound links, and regular keywords. Regular browsers search the surface Web, but that is where the search stops. To explore more, let us understand that Web content is available for search in three models:

- Free content that can be searched by popular search engines
- Content that is accessible on a payment basis
- Content that is accessible by logging in to the website

Based on access (using popular search engines such as Bing, Google, or Yahoo), we categorize the Web into three categories:

- **Open/Surface Web**: Open Web is a small part of the Web that be easily accessed using popular search engines.
- **Deep Web**: The password-protected Web, or paid services, subscription services such as Netflix, databases (academic databases), or Web pages that can only be accessed through an online form (email services) forms the deep Web. This may directly not be accessible through popular search engines but is accessible if the link is known.
- **Dark Web**: Part of the Web that has hidden IP addresses and cannot be directly accessed through popular search engines is known as Dark Web. Accessing such sites requires specific tools and is hidden from popular search engines. It is said that most of the dark Web is engaged in illegal activities such as Credit Card details data, Weapon selling websites, Drug selling websites, Selling of stolen products, and so on.

In this section, we will focus on the deep Web. Deep Web is the information that is reachable over the Web; it is dynamically available in response to queries and is not placed on static pages ahead of time. Recent studies indicate that the deep Web has hundreds of times more data than the surface Web. Only a very small part of Web pages is indexed, and a major part has not been indexed. So, we can understand that **the deep Web, invisible Web, or hidden Web is a part of the worldwide Web, the contents of which are not indexed by standard search engines for some reason.** Once they get indexed, they will also become part of the open/ surface Web. The deep Web is a massive repository of information, which can be reached only through specific tools (like selenium), not through regular Web crawlers. For example:

Some of the examples of the deep Web are as follows:

- Content of your email id
- Content of your social media account
- Content of online banking account
- Content of the companies that is stored in the databases
- Legal documents
- Medical records
- Content contained in scientific and academic databases (for example: academic journals)

We have seen how to scrape the open Web in *Chapter 5, Web Scraping*. It will be difficult to scrape the following type of sites (deep Web):

- Proprietary sites
- Sites that need registration
- Dynamic sites
- Unlinked pages
- Sites that are blocked by local Webmasters
- Sites that are blocked by search engine policy
- Sites with specific special formats
- Searchable databases
- Private Web, and so on.

Proprietary sites generally require a fee if you want to crawl them. As for registration sites, they require a login-id and password. Dynamic websites' data is created on demand and has no existence before the query and limited existence afterward. If you ever noticed an interesting link on a social media site or on a news site but found that the link was inaccessible later on, then you have encountered an ephemeral website.

However, the most valuable deep-learning resources are available in the form of

searchable databases. There are a huge number of secure databases with information worth billions. But they are all mostly un-scrapable. They serve as the back-end to front-end search bars in various sites- Sites that will let you view a part of the data at one go but never the whole.

Many individuals and institutions are digging up this information and are making it available to all (trying to get it indexed). For example, OAIster (pronounced "oyster") is an online combined bibliographic catalog by The University of Michigan that provides open-access material. This catalog reveals lots of scholarly resources, which have been harvested using the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH). Many other Web pages provide information, links, and references collected from the Deep Web.

Note:

1. Deep Web and Dark Web are different.
2. The rule of thumb is that if you need to log in using a username and password or any other type of authentication, you are accessing a deep Web.
3. Some other examples of the Deep Web catalog are https://www.findarticles.com/, The Library Spot (www.libraryspot.com), UCLA online Library, www.infoplease.com, and many others.

We have already discussed how scraping can be done on open Web pages and deep Web pages, that is, pages with login or database pages (see *Chapter 5, "Web Scraping"*). As we learned that deep Web pages might be accessed using search engines once we know the link to the page, login password, or access to the database. To access deep Web pages, the only thing that will differ is to provide login credentials to the scraping scripts and then follow the same process of graph analysis that is used by surface Web pages. Link analysis of the deep Web can also be done using graph analysis (as discussed in the previous section and upcoming chapter).

Web Search and Hyperlinks

The worldwide Web is a collection of Web pages stored on computers, connected to the internet (often termed as Web). Web pages contain information in the form of text, images, and videos. They also provide links to some other part on the same Web page or other Web page using their URL. These links are called hyperlinks and are the easiest way for navigation across the Web.

The Web has now become a very resourceful entity in terms of information and is becoming a de facto database for search for information. Searching information from the Web is often termed as Web search and may be understood as the search of information, images, video, multimedia content, and so on, on the Web using a search engine.

Web pages are interlinked with each other using hyperlinks. The hyperlink is a text or image that is a reference to the other Web page. The hyperlink is usually blue and, when clicked, jumps to the referenced Web page. The hyperlink structure can be used for hyperlink analysis and can be defined as an area of Web information retrieval using the Web structure. The two main uses of hyperlink analysis in Web information retrieval are crawling and ranking. Other uses of hyperlink analysis include computing the geographic scope of a Web page, finding mirrored hosts, and computing statistics of Web pages and search engines.

Hyperlink analysis on the Web

Most search engines use hyperlink analysis algorithms to deliver focused results to user queries. Hyperlink analysis algorithms make either one or both of the following simplifying assumptions:

Assumption 1. A hyperlink from page A to page B is a recommendation of page B by the author of page A.

Assumption 2. If page A and page B are connected by a hyperlink, then they might be on the same topic.

Millions of pages may contain what you are trying to search for, so to provide the most relevant results, some algorithms have been designed. The algorithms try to compute the importance of a Web page (node in a graph) using hyperlink-based algorithms PageRank and HITS. These algorithms were created in 1998 and are used for ranking Web pages. The algorithms consider the Web as a complex graph that uses the concepts of social network analysis, as discussed in *Chapter 8*, "Social Network Analysis using Python").

These algorithms use hyperlink structure to give rank according to the degree of prestige or authority. One of the most popular search engines, Google, uses the PageRank algorithm, which was developed by Google founders Sergey Brin and Larry Page in 1998. This algorithm provides a ranking to each Web page based on how many links are leading to that page from other sites (Inlinks) (for details, see *Chapter 2, "Web Mining Taxonomy"*). The concept behind the PageRank algorithm is that if a random Web surfer on the Web visits a page and follows the links keep visiting linked pages (random walk). The popularity of any page is computed as the average number of times this random visitor has visited a particular page following the links. The probability that a page is visited by a random surfer on the Web is considered an important factor for ranking the search results. This probability is termed as page-rank, which is computed iteratively.

Consider *figure 7.7*; note that it has 6 Web pages that are connected with each other using links (represented as the directed graph).

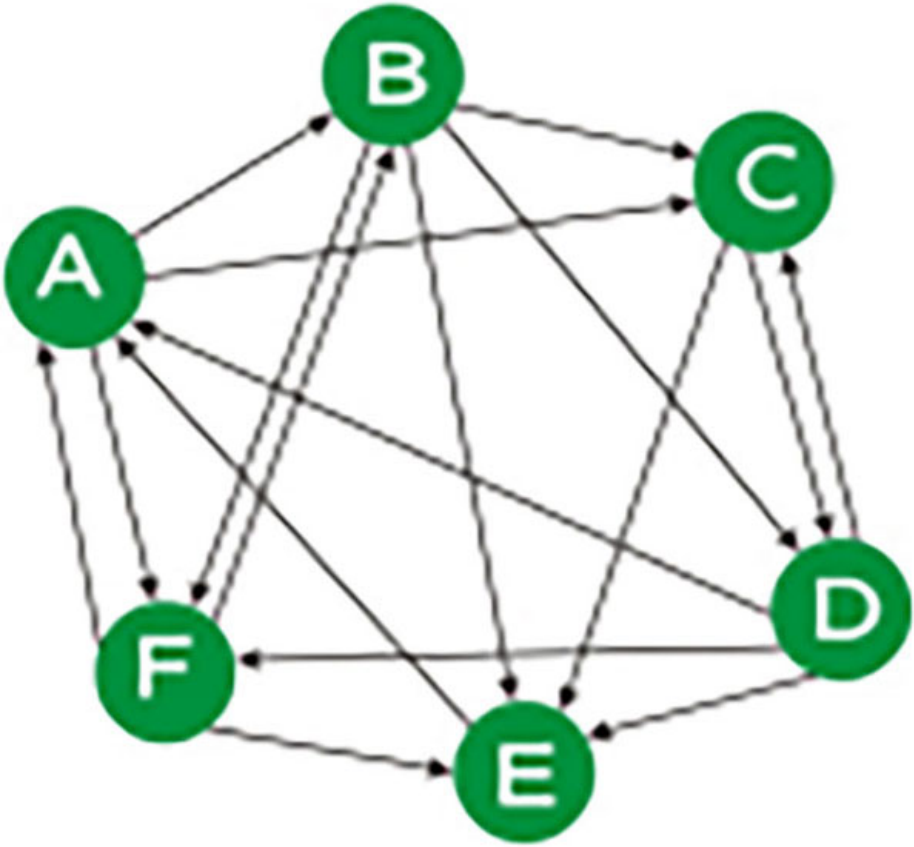


Figure 7.7: *Sample small Web with six pages*

The number of times each page is visited through links from other pages (shown by Inlinks) is as follows:

Page A: 3

Page B: 2

Page C: 3

Page D: 2

Page E: 4

Page F: 3

Thus, the rank of page E, which is 4, is the highest, and thus, we can infer that Page E is the most visited page (see *Chapter 2, "Web Mining Taxonomy"*, to understand how the PageRank algorithm works).

When PageRank was being developed, Jon Kleinberg, a professor in the Department of Computer Science at Cornell, came up with a parallel solution for the Web search problem. He developed an algorithm that is based on the link structure of the Web to discover and rank the pages relevant for a particular search. This algorithm is named as Hyperlink Induced Topic Search (HITS) algorithm.

HITS is a link analysis algorithm that uses a Web-link structure to discover and rank the Web pages relevant for a particular search, based on hubs and authorities (*Chapter 2, "Web Mining Taxonomy")* to define the recursive relationship between Web pages. This algorithm treats the world wide Web as a directed graph G(V,E) where V is a set of vertices representing pages and E is a set of edges corresponding to the link. He added a human touch to the Web search.

Both these algorithms make use of the link structure of the Web graph to decide the relevance of the page. The difference between them is that HITS operates on a small subgraph from the Web graph. This subgraph is query-dependent, so whenever we change the query, the subgraph changes.

Hyperlink Induced Topic Search (HITS)

In this section, we will focus on the HITS algorithm, which is one way of ranking the Web pages that are relevant for a particular Web search; another is PageRank (see *Chapter 2, Web Mining Taxonomy)*, which talks about the concept of link analysis to understand how a search engine finds the most relevant pages based on the given string.

Given a query to a search engine, the HITS algorithm uses the authority score (that is assigned to authorities) and hub score (assigned to authorities) on each Web page. In 1999, Kleinberg suggested that there are two types of pages that are relevant to the query: the pages that contain relevant information about the query, known as ***authorities***, and pointers to good information sources, known as ***hubs*** (*figure 7.8*). Both types of such pages are connected to each other in a way that good hubs (that contain query strings) point to many good authorities, and many good authorities point to good hubs. Web pages may have a double identity, as a hub for some pages and authority for others.

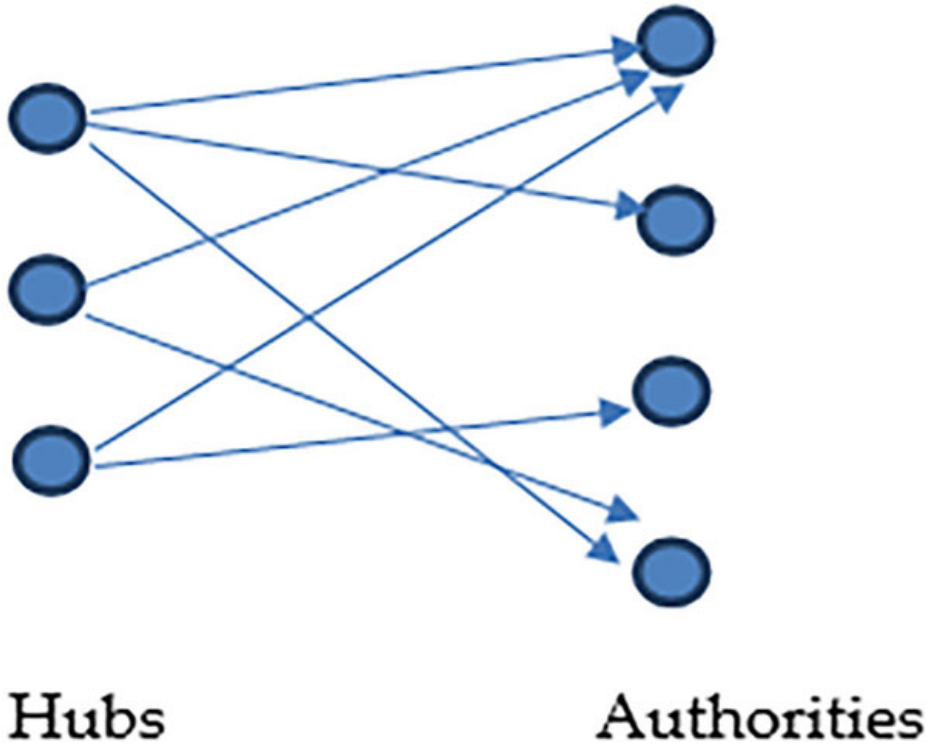


Figure 7.8: *Hubs and Authorities*

According to Ding et al. and Kleinberg, a good hub page is a page that is pointing to many authority pages on that content; similarly, a good authority page is a page that is pointed to by many hubs. Any page can be a good hub as well as a good authority at the same time. To find the most relevant pages, Kleinberg suggested to associate each page with a hub score and an authority score. Considering N as the total number of pages connected to "p" and "i" is a page connected to "p". For page p with a hub score H(p) and an authority score A(p) these scores are computed iteratively as follows:

- Assign each node/Web page a hub and an authority a score of 1.
- **Apply the Authority Update Rule**: each node's **authority score** is the **sum of hub scores** of each node that **point to it**.

$$\text{auth}(p) = \sum_{i=1N} hub(i)$$

- **Apply Hub Update Rule:** each node's **hub score** is the **sum of the authority scores** of each node that **it points to**.

$$\text{hub}(p) = \sum_{i=1N} auth(i)$$

- Normalize Authority and Hub score as follows:

  auth(j) =auth(j)/$\sum_{i=1N} auth(i)$**,** that is, to normalize the authority score of node "j", divide the authority score of "j" by the sum of the authority scores

of all nodes.

- Repeat k times

Let us understand this with the help of an example. Consider the following graph that represents a subgraph of a network:

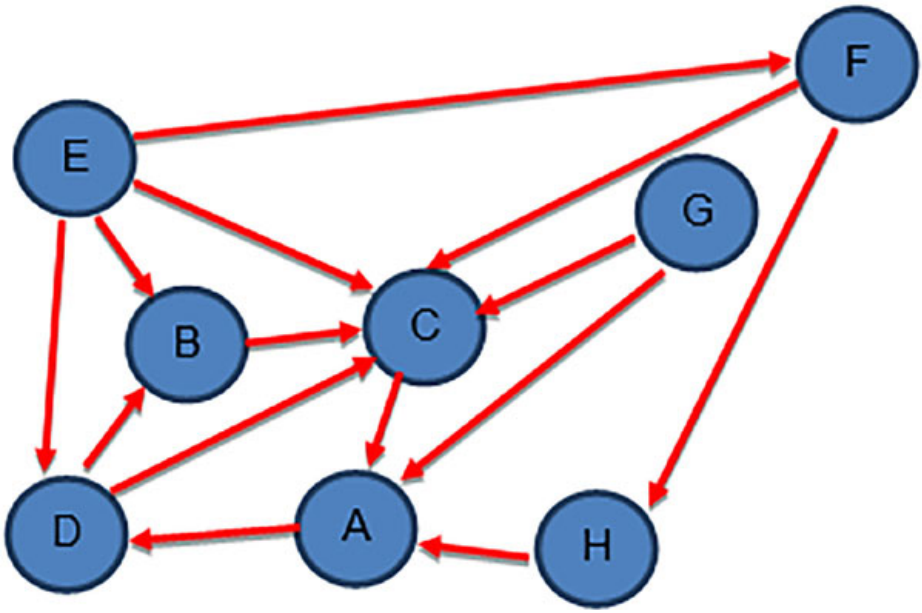


Figure 7.9: *Subgraph of a network*

Considering the preceding subgraph (*figure 7.9*), we now create a matrix for the Hub score and authority score. As discussed, we assign each hub and authority with an initial score of 1 (Step 1). We then traverse the graph and assign hubs and authorities a new value based on In-links and Out-links. Beginning with node "A", we see that the number of In-links to node "A" are 3, and out link is only 1. Thus, we assign "3" as the authority score and "1" as the hub score to node "A"; for node "B" it is 2 and 2; For the node C it is 5 and 1 and so on. After traversing all nodes, we get the final scores as follows (Steps 2 and 3):

Table 7.1: *Matrix of Hub and Authority score*

Then as per Step 4, we need to normalize these scores. Thus, we find the sum of hubs and authorities, which is computed as follows:

$\sum i€N \text{ auth}(i)=15$

$\sum i€N \text{ hub}(i)=15$

We then compute the normalized scores as follows:

Table 7.2: *Normalized Authority and Hub scores*

Now, we move to the next iteration, where the new hub and authorities become an old hub and authority scores as shown. Now, we need to compute the new Hub and

Authority scores by applying Hub and Authority Update Rules. It is important that the scores are not 1 now, and we must compute the scores with the new values. For node "A", we have nodes "C", "G", and "H" pointing to "A" and "A" is pointing to "D". We now find the sum of the hub scores of "C", "G", and "H" (as per Step 2) to compute authority scores for node "A". It is computed as follows:

$$Auth(A)=1/15 + 2/15 + 1/15 =4/15 \text{ (Sum of hub scores of 'A')}$$

$$Hub(A)=2/15 \text{ (Sum of authority scores of 'A')}$$

For node "B" has two nodes, "D" and "E" pointing to it, and node "B" is pointing to 2 nodes, "C" and "E", so the computation of new scores will be computed as follows:

$$Auth(B)=2/15 + 4/15 =4/15$$

$$Hub(B)=6/15$$

And for node "C", it is computed as follows:

$$Auth(C)=2/15 + 2/15 + 4/15 +2/15 + 2/15=12/15$$

$$Hub(C)=3/15$$

Computing in a similar manner, we get the following matrix for the normalized scores of this iteration:

Table 7.3: *Matrix for the normalized scores from the next iteration*

If we continue to iterate again and again, we get the final values as follows. For k=2, as we have computed just now, the final figures are:

Table 7.4: *Final normalized scores at the end of the second iteration*

Notice that the value of the authority score and hub score is changing for some nodes; for others, they are not changing. As the number of iterations increases in a larger network, the Hub scores and Authority scores converge to a unique value. As per various research conducted, it is said that typically these values converge between 5 and 6 iterations. So, considering the values of the 6th iteration, we have the following authority and hub scores:

Table 7.5: *Final normalized scores at the end of all iterations (6th iteration)*

Notice that the authority score of nodes "B" and "C" is the highest, and the hub score of nodes "D" and "E" is the highest. Thus, we can say these nodes are the most relevant nodes with respect to query results.

HITS algorithm can be implemented using the Python library NetworkX. NetworkX provides function hits(G), where G is the input graph for which the authority and hub scores are to be computed. hits(G) return two dictionaries that contain hub and authority scores of graph G.

HITS algorithm combines the content-based search with link-based search. It makes the basic assumption that if the pages from the root set are closed to the

query topic, the pages belonging to the base set (one link farther) are, by their content, similar to the query. An important difference between PageRank and HITS is the way that page scores are propagated in the Web graph. As we have seen that the same pages may be hubs and authorities at the same time, sometimes In some cases, HITS may not produce the most relevant documents to the user queries because of equivalent weights as HITS gives equal importance to automatically generated links that may not have relevant topics for the user query. Though theoretically, HITS may be a very good algorithm, it is not efficient in real-time.

The main drawback of this algorithm is that the hubs and authority score must be computed iteratively from the query result, which does not meet the real-time constraints of an online search engine. The complexity of the PageRank algorithm is O(log N), whereas the complexity of HITS algorithms are less than O(log N). Thus, it could not be implemented in real-time search engines.

Partitioning algorithm

As we have discussed, the Web is considered as a graph with nodes and edges (vertices). Graph partitioning techniques are used to distribute the whole graph as a disjoint subset to a different device. The benefit of such partitioning and distributing huge graph data set is to process data efficiently and faster process of any graph-related applications. A good partitioning algorithm is good for any data-intensive application and always aims to reduce the communication between a machine in their distributed environment and distribute vertices roughly equal to all machines. It is very helpful in processing complex networks like social networks (*Chapter 8, "Social Network Analysis using Python"*), biological networks, and transportation networks. It also plays an important role in the PageRank algorithm that is used to compute the rank of the Web rank from the Web network (*Chapter 2, "Web Mining Taxonomy"*. Finding a rank and importance of a Web page from a Web graph would be effective by partitioning the whole graph into several distributed machines.

Various algorithms are available in the areas of graph partitioning and graph clustering. These graph partitioning methods can be used to speed up graph searches.

One of the historical graph partitioning techniques is the technique as proposed by Kernighan-Lin (Michelle Girvan and Mark Newman).

Girvan and Newman applied the concept of finding edges in a network that occur most frequently between other pairs of nodes by finding edges betweenness centrality. It is expected that the edges joining communities will have a high-edge betweenness. The underlying community structure of the network will be much more fine-grained once the edges with the highest betweenness are eliminated, which means that communities will be much easier to spot.

Girvan–Newman algorithm is a hierarchical method used to detect communities in a graph depending on the iterative elimination of edges with the highest number of

shortest paths that go through them. Essentially, in this algorithm, the process of repetitive calculations of edge betweenness centrality of all nodes and cutting the edge with the highest centrality.

The Girvan–Newman algorithm can be divided into four main steps:

For every edge in a graph,

1. Calculate the edge betweenness centrality of all existing edges.
2. Remove the edge with the highest betweenness centrality.
3. Recalculate the betweenness centrality for every remaining edge.
4. Repeat Steps 2–4 until there are no more edges left. Connected nodes are considered communities

Between centrality is a measure of the node falling in the shortest paths of other pairs of nodes. As the Nodes and edges with high betweenness have more control over information passing between others, thus, may have considerable influence within a network. There are many ways to compute betweenness centrality, and any one method is not standardized. There are many ways to solve it**. "***It is also defined as the number of shortest paths in the graph that passes through the node or edge divided by the total number of shortest paths, and is one of the most widely used shortest-path-based centrality measures for complex network analysis."*

Thus, to compute between centrality, we need to compute the shortest path. Let us consider the following network graph as follows (*figure 7.10*):

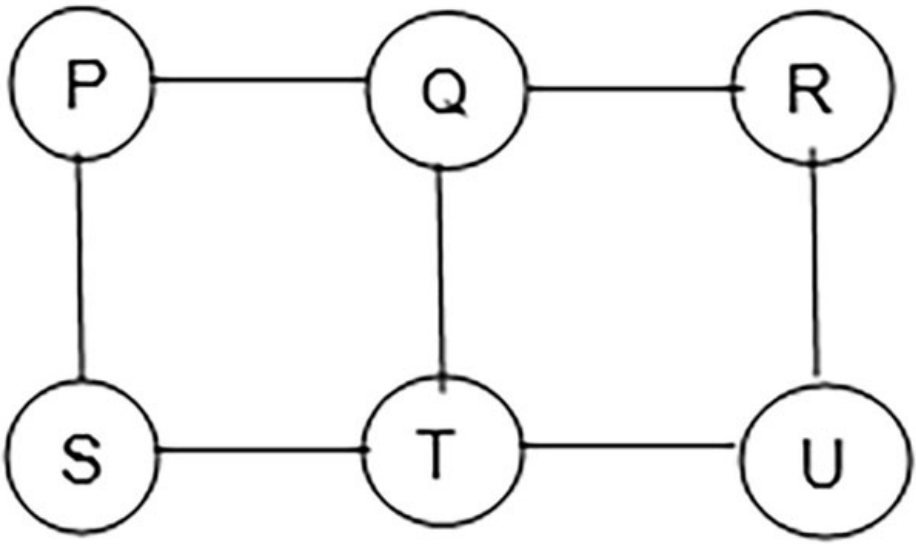


Figure 7.10: *Example network graph*

In this graph, we have 6 nodes and 7 edges. This graph can also be considered as follows:

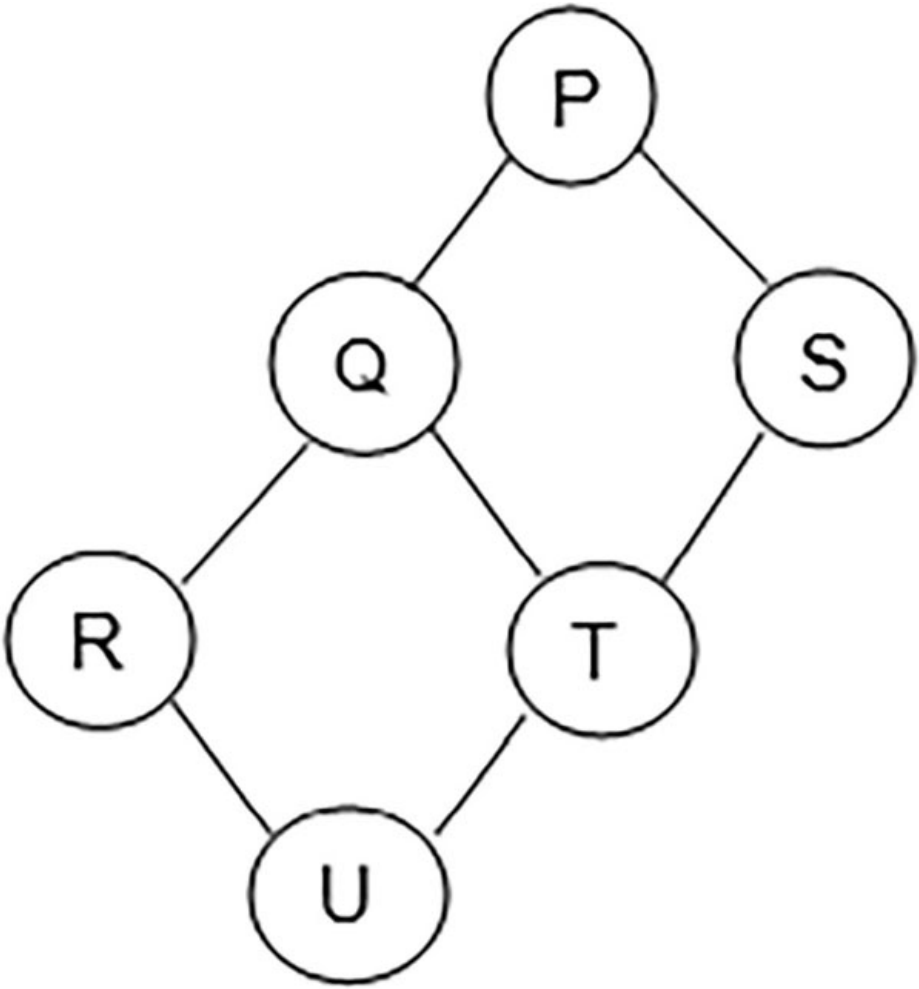


Figure 7.11: *Network shown as graph*

To implement the Girvan–Newman algorithm, the most important step is the computation of the shortest path of each node (we use **Breadth First Algorithm** (**BFS** for the same). So, the shortest path can be computed as follows:

Number of shortest Path to Q : P->Q =1

Number of shortest Path to S: P->S=1

Number of shortest Path to T : P->Q->T, P->S->T =2 (Sum of Q=1 and S=1)

Number of shortest Path to R : P->Q->R =1

Number of shortest Path to U: P->Q->R->U, P->Q->T->U, P->S->T->U =3 (Sum of R=1 and T=2)

In larger networks, it is not possible to compute it node by node, so in general, we compute the shortest path to any node as the sum of the shortest path of the parent nodes. We can affirm this from the following figure:

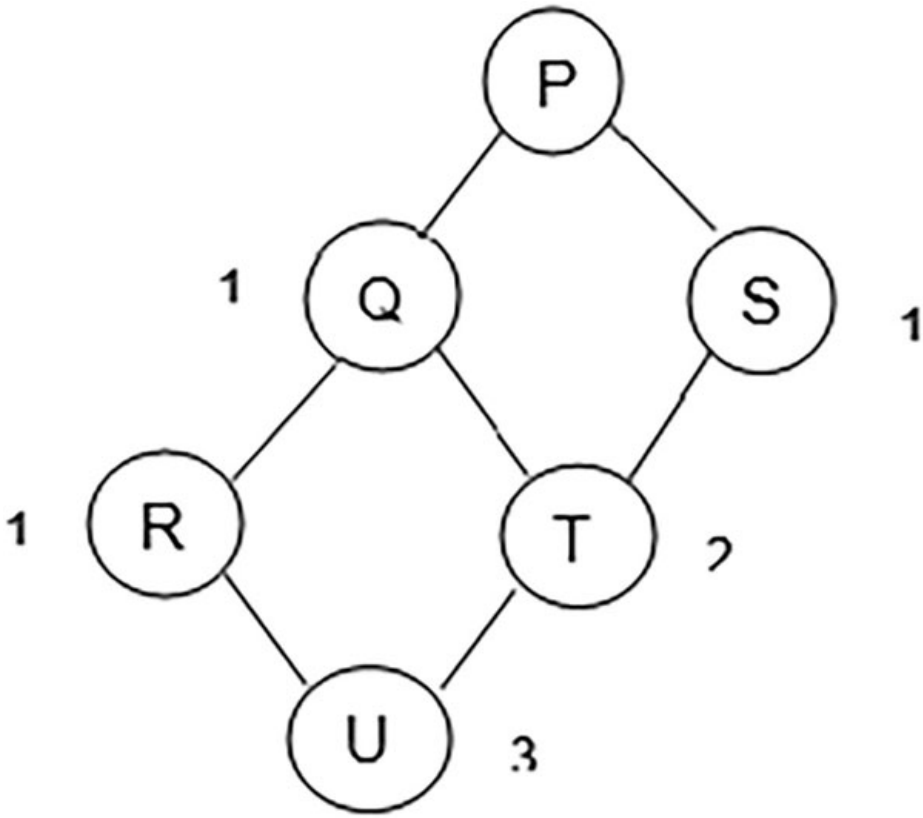


Figure 7.12: *The shortest path of each node*

After the computation of the shortest path, the next step is to find the edge weight/ value of each edge on the graph (that represents a fraction of the shortest paths between any two nodes going through the node). For this, we start from the bottom of the converted graph (*figure* 7.12) and divide the preceding node's value by the bottom node's value, getting a fraction of the value. From there on out, the edge values are computed as follows:

Edge weight/ value= Previous edge value + the value of node that leads into it/ the previous node value

We first compute the score for the edges UR and UT, the first level edges in the backward direction.

Edge score of UR = Ratio of score of R/ Score of U = 1/3 =0.33

Edge score of UT = Ratio of score of T/ Score of U = 2/3 =0.667

We first compute the score for the next level of edges in the backward direction

Edge score of QR = (Previous edge (UR) value + the value of node that leads into it (Q)) / the previous node value (R)= (0.33+1)/1 =1.33

Edge score of QT = (Previous edge (UT) value + the value of node that leads into it

(Q)) / the previous node value (T) = **(0.667+1)/2 =0.835**

Edge score of ST = (Previous edge (UT) value + the value of node that leads into it (S)) / the previous node value (T) = **(0.667+1)/2 =0.835**

This process will be iterated for each previous level of edges up to the root. In this graph, we just have more levels of edges, that is, PQ and PS:

Edge score of PQ = (Previous edge (QR + QT) value + the value of node that leads into it (P))/ the previous node value (Q) = (1.33+0.835+1)/1 =3.165

Edge score of PS = (Previous edge (ST) value + the value of node that leads into it (P)) / the previous node value (Q)= (0.835+1)/1=1.835

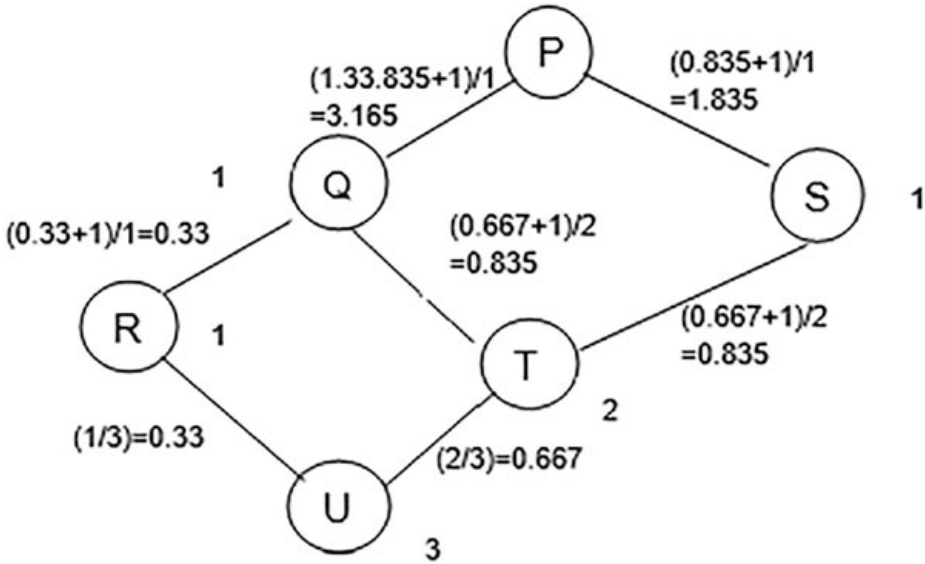


Figure 7.13: *Calculating between centrality with Node P*

In this step, the edge scores of the shortest paths with respect to node A have been computed. We will now repeat the same steps from the other remaining five nodes (Q, R, U, T, and S). In the end, we get a set of six scores for all the edges in the network. After adding these scores, they are assigned to the original graph as shown follows:

Edge	Edge betweenness (Obtained by adding score obtained by computing shortest path with respect to each node)						
	Node P	Node Q	Node R	Node S	Node T	Node U	Sum (P+Q+R+S+T+U)
PQ	3.165	1.5	1.33	0.835	0.5	0.667	8
PS	1.835	0.5	0.33	1.835	0.5	0.5	5.33
QR	3.165	1.5	1.33	0.835	0.5	0.667	8
QT	0.835	2	0.835	0.835	2	0.835	7.34
RU	1.835	0.5	0.33	1.835	0.5	0.33	5.33
ST	3.165	1.5	1.33	0.835	0.5	0.67	8
UT	3.165	1.5	1.33	0.835	0.5	0.67	8

Table 7.6: *Edge betweenness scores of shortest paths with respect to each node*

So, the graph can be seen as follows:

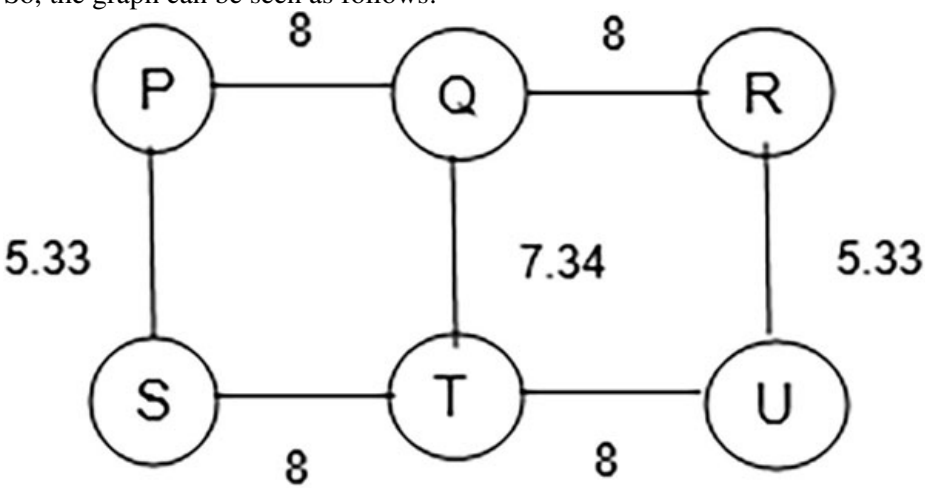


Figure 7.14: *Graph with edge scores*

Since this is a bidirectional graph, we divide these values by 2. The graph with new values is shown as follows:

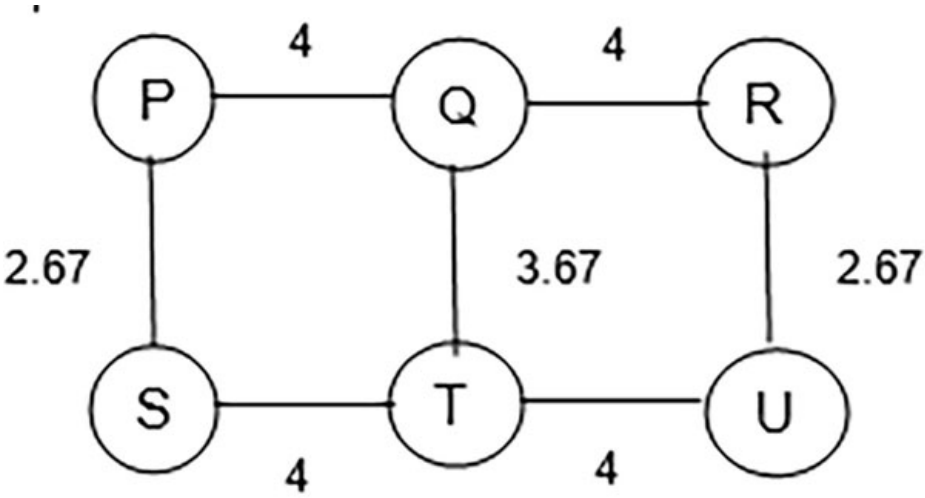


Figure 7.15: *Graph with new edge scores (previous score divided by 2)*

We can see the four edges, PQ, QR, ST, and UT, have the same weight, which is the highest weight. Girvan–Newman algorithm says to remove the edges with the highest weight, so we remove these edges. We get the following graph:

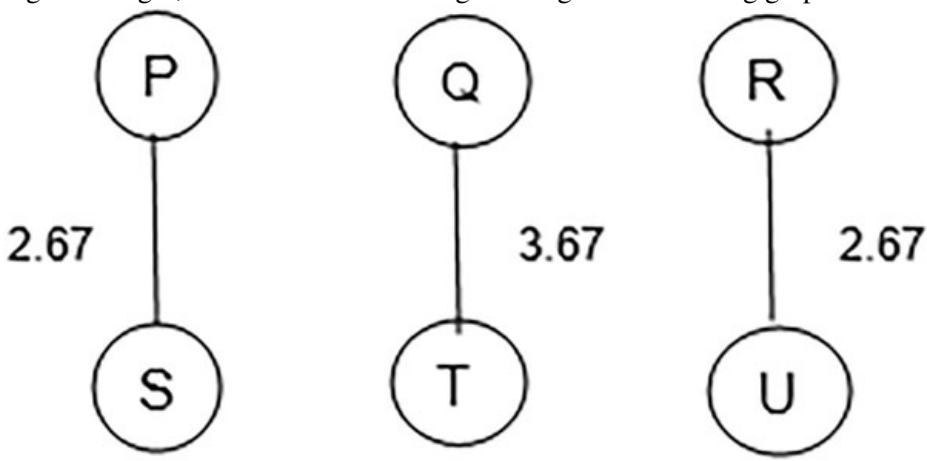


Figure 7.16: *Graph after removing edges with the highest edge score*

In this graph, we get three subgraphs of the original graph, and if we break it further, we will get independent nodes P, Q, R, S, T, and U; thus, this graph cannot be further decomposed. So, as per the Girvan–Newman algorithm that says once we reach the stage when the graph cannot be further decomposed, we get the communities. So, in this example, we get three communities as PS, QT, and UR.

Implementation in Python

To understand how graphs can be used for network analysis using the graph, we will see a small example where we can implement link analysis using the graph.

We take an example of community detection in a social network (*Chapter 8, Social Network Analysis using Python*), which could also be implemented on related websites to find a community of linked Web pages. Think about any social media platform, such as Facebook, Instagram, or Twitter, where several people are connected. This consists of various social circles like relatives, schoolmates, colleagues, friends, people working in similar domains, and so on. These social circles are kind of communities.

Detecting communities in a network is one of the most important tasks of network analysis. This task becomes very difficult if done manually; thus, we use community detection algorithms that are based on graphs to partition the network into multiple communities. One of the most such algorithms is the Girvan–Newman algorithm (a divide-based method) that is used for the detection and analysis of community structure and relies on the iterative elimination of edges having the highest number of shortest paths between nodes passing through them. Removing edges one by one from the graph, the network breaks down into smaller pieces, so-called communities.

For larger networks, it is not possible to compute the edge between centrality manually. So, we use the Girvan–Newman algorithm. Pseudocode for the same is given as follows:

The pseudocode used to process the algorithm is given as follows:

```
REPEAT
LE Th BEnumbeorfedgeisnthegraph
FORi = 0ton - 1
LETB[i]BEbetweenmceesnstralcifteydgé
IFB[i]} max_TBHEN
max_=BB[i]
max_B_e=digé
ENDIF
ENDFOR
removeedgé fromgraph
UNTInLumbeorfedgeisngrapihs0
```

We implement the same using Python `Networkx` given:

```
1.impomtatplotlib.apsyppllto t
2.impontetworaksxnx
3.frometworkx.algorithms.commuinmiptoyr.t
girvan_newman
4.
5.G = nx.karate_club_graph()
6.communit=igeisrvan_newman(G)
7.
8.node_gr=u[p]s
9.forcominnext(communities):
10.  node_groups.append(list(com))
```

```
print(node_groups)

color_map = []
for node in G:
    if node in node_groups[0]:
        color_map.append('blue')
    else:
        color_map.append('green')
nx.draw(G, node_color=color_map, with_labels=True)
plt.show()
```

Output:

```
[[0, 1, 3, 4, 5, 6, 7, 10, 11, 12, 13, 16, 17, 19, 21],
[2, 8, 9, 14, 15, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33]]
```

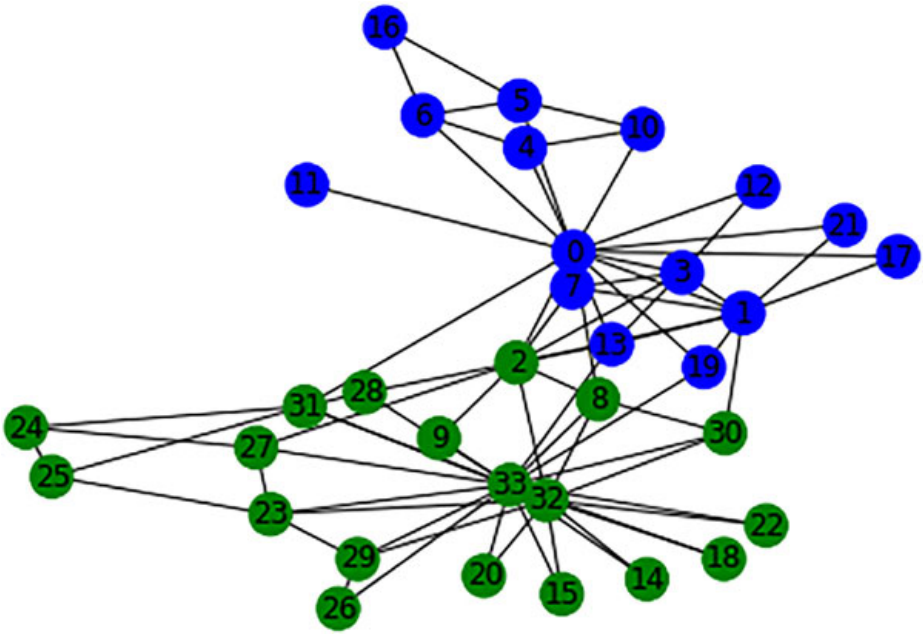


Figure 7.17: *Graph showing two communities using Girvan–Newman algorithm*

Conclusion

The structure of a Web typically consists of Web pages and hyperlinks connecting them. Structure mining is used to generate the structural details about the website. It discovers the link structure of the hyperlinks within the Web. Web structure mining helps us to identify similar websites, classify the website, to establish the relationship between websites, or discover websites. This chapter begins with a discussion of the concept of Web structure mining. It then discusses about the

structure of the Web page that discusses that the link within the Web page can be used to connect to any other location of the Web page or any other Web page of some other website. This acts as a connection point between two pages and can be a text or an image. **Web structure mining** is the process to extract patterns from hyperlinks on the Web. It is also called **link mining**. After this, the types of Web structure mining are discussed, with examples. It then discusses about Web graphs and how Web graphs can be used to represent Web page structure to perform Web graph mining. Link Analysis used for Web-based Information Extraction focuses on possible link detection between concepts across multiple documents. The chapter focuses on Web information extraction. A brief introduction to deep Web mining and how to extract information from such pages is also discussed. The chapter then discusses about Web search and the algorithms related to Web search. One of the algorithms, Hyperlink Induced Topic Search (HITS), is discussed in detail with a worked example. A partition algorithm is discussed that is used to optimize analysis, and a case study demonstrates the concepts studied in the chapter. Link Analysis has been implemented using NetworkX in Python. To understand **implementation of social network analysis, the Girvan–Newman algorithm** is used to **detect communities** and is implemented using a Python program.

In the upcoming chapter, we will discuss about social networks using Python. **Social Network Analysis** (**SNA**) is the process of investigating social structures through the use of networks and graph theory. It characterizes networked structures in terms of nodes (individual actors, people, or things within the network) and the ties, edges, or links (relationships or interactions) that connect them. This becomes very useful as we have a lot of social networks such as Twitter, Facebook, and many others.

Points to remember

- **Weblink:** The link or reference of the Web page through which a page is opened, irrespective of on which Web server it is stored in.
- **Hyperlink:** The link within the Web page that can be used to connect to any other location of the Web page or any other Web page of some other website. This acts as a connection point between two pages and can be a text or an image.
- **Web structure mining** is the process to extract patterns from hyperlinks on the Web. It is also called **link mining**.
- **Link-based Classification:** Link-based classification is the task of classifying Web pages using the relations or links present amongst them.
- **Link-based Cluster analysis:** In clusters-based analysis, the data/ Web pages is/are grouped into various clusters where similar objects are grouped together, and dissimilar objects are grouped in a different group.
- **Link Type analysis:** Link type analysis is involved with the analysis of the

different type of links, that is, the internal links (links within the same domain), external links (links in an outside domain), and back links (the links that bring back to the domain from where you left to visit the external domain).

- **Backlinks** are links from one page on one website to another. If someone links to your site, then you have a backlink from them. If you link to another website, then they have a backlink from you.
- Search engines like Google see backlinks as votes of confidence. In general, the more votes your Web pages have, the more likely they are to rank for relevant search queries.
- **Link Strength:** Link Strength is a metric to calculate **the overall strength of backlinks**. The Link Strength is indicated on a scale from 0 to 100. The higher the score, the more powerful the backlink is.
- **Link Cardinality:** Cardinality indicates the number of occurrences (one or many) that one entity has relative to another.
- The Web is represented in the form of a graph as in mathematics, and relationships can be expressed in terms of a graph.
- A **graph** is depicted in diagrammatic form as a set of dots for the vertices (or nodes) that are connected by lines or curves for the edges.
- **Graph** is also considered as a collection of nodes and edges that represents relationships between nodes and edges.
- The graph edges may have a **weight** that indicates the strength of the connection between the objects.
- **Web Graph Mining** Set of tools and techniques used to analyze the properties of real-world Web graphs to predict how the structure and properties of a given Web graph might affect some applications and develop models that can generate realistic graphs that match the patterns found in real-world Web graphs of interest.
- **Information Extraction (IE)** is the process of automatically extracting a structured piece of information from structured or semi-structured text documents using various tools and techniques.
- **Web information extraction** is the application of IE techniques to process the vast amount of unstructured content on the Web.
- **Web information extraction** may be **used to** extract reviews, opinions, and sentiments also, along with the named entity and relationship extraction.
- **Entity Extraction** refers to the identification of mentions of the named entities (such as the name of a person, location, organization, and so on.) from the unstructured content on the Web.
- **Relationship extraction** is used to create structured knowledge bases from unstructured text or Web pages. This type of extraction deals with associating pairs of named entities.

- **Sentiment Analysis or Opinion Mining** is the technique to reliably extract reviews, opinions, and sentiments about various products and services from the content present on various online Web pages.
- **Open/ Surface Web**: Open Web is a small part of the Web that be easily accessed using popular search engines.
- **Deep Web:** The password-protected Web, or paid services, subscription services such as Netflix, databases (academic databases), or Web pages that can only be accessed through an online form (email services) forms the deep Web. This may directly not be accessible through popular search engines but is accessible if the link is known.
- **Dark Web**: Dark Web is the part of the Web that has hidden IP addresses and cannot be directly accessed through popular search engines. Accessing such sites requires specific tools and is hidden from popular search engines. It is said that most of the dark Web is engaged in illegal activities such as Credit Card details data, Weapon selling websites, Drug selling websites, Selling of stolen products, and so on.
- Searching information from the Web is often termed as **Web search.**
- The **hyperlink** is a text or image that is a reference to the other Web page.
- **PageRank algorithm** was developed by Google founders Sergey Brin and Larry Page in 1998. This algorithm provides a ranking to each Web page based on how many links are leading to that page from other sites (Inlinks).
- **HITS algorithm** uses the authority score (that is assigned to authorities) and hub score (assigned to authorities) for each Web page.
- **Partitioning Algorithm** is good for any data-intensive application and always aims to reduce the communication between a machine in their distributed environment and distribute vertices roughly equal to all machines.
- **Link Analysis** can be implemented using NetworkX in Python.
- To understand **implementation of social network analysis**, **Girvan–Newman algorithm** is used to detect communities and is implemented using a Python program.

MCQs

1. Web structure mining is also known as:

    a. Web Analysis

    b. Link Analysis

    c. Structure Analysis

    d. None of above

2. A link used to point to another location on the same page or on a different page is known as:

a. Web Link
b. Hyperlink
c. Both
d. None of above

3. _____ is a metric to calculate the overall strength of backlinks:

a. Link Cardinality
b. Backlinks
c. Link Strength
d. None of above

4. _____ is used to represent Webpages and their links in the form of nodes and edges:

a. Graph
b. Backlinks
c. Link Strength
d. None of above

5. Web information extraction is used to extract:

a. Reviews
b. Opinions/Sentiments
c. Named entity and relationships
d. All of above

6. Web is categorized as:

a. Open Web, Deep Web, Dark Web
b. HTML documents and Web pages
c. Only a
d. Both a and b

7. PageRank algorithm was developed by:

a. Sergey Brin and Larry Page
b. Girvan-Newman
c. Guido van Rossum
d. None of these

8. PageRank algorithm is based on:

a. Providing authority and hub score to Web pages
b. Providing ranking to each Web page

c. Both of these

d. None of these

9. HITS algorithm is based on:

   a. Providing authority and hub score to Web pages

   b. Providing ranking to each Web page

   c. Both of these

   d. None of these

10. Which of the following type of Web is not easily accessible?

    a. Deep Web and Dark Web

    b. Open Web and Deep Web

    c. Open Web and Dark Web

    d. None of these

Answers

Questions

1. What do you understand by Web structure mining? Write about different types of Web structure mining.
2. Web pages are represented in the form of a graph with nodes and edges. Explain this statement with the help of an example.
3. Define information extraction. How is it relevant to the Web?
4. Define the following terms:

   a. Graph

   b. Deep Web

   c. Open Web

   d. Dark Web

   e. Hyperlink

5. How is mining a dark Web different from mining a deep Web? Give an example to explain.
6. Elaborate on the use of PageRank algorithm and HITS algorithm used for the Web search.
7. Explain PageRank, its importance, and how it factors into search engine optimization (SEO).
8. What are hubs and authorities in the HITS algorithm? What purpose do they serve?

9. Explain the difference between PageRank and HITS algorithms.
10. What is the use of the partitioning algorithm?

Key terms

- ***Weblink*** is the link or reference of the Web page through which a page is opened, irrespective of on which Web server it is stored.
- ***Hyperlink*** is a link within the Web page that can be used to connect to any other location of the Web page or any other Web page of some other website.
- ***Web structure mining*** is the process to extract patterns from hyperlinks on the Web. It is also called **link mining**.
- ***Backlinks*** are incoming links from some other website to that page.
- *Link Strength* is a metric to calculate the overall strength of backlinks.
- ***Link Cardinality*** indicates the number of occurrences (one or many) that one entity has relative to another.
- ***Graph*** is depicted in diagrammatic form as a set of dots for the vertices (or nodes) that are connected by lines or curves for the edges.
- ***Web Graph Mining*** is the set of tools and techniques used to (a) analyze the properties of real-world Web graphs, (b) predict how the structure and properties of a given Web graph might affect some application, and (c) develop models that can generate realistic graphs that match the patterns found in real-world Web graphs of interest.
- ***Information Extraction (IE)*** is the process of automatically extracting a structured piece of information from structured or semi-structured text documents using various tools and techniques.
- ***Web information extraction*** is the application of IE techniques to process the vast amount of unstructured content on the Web.
- ***Entity Extraction*** refers to the identification of mentions of the named entities (such as the name of a person, location, organization, and so on) from the unstructured content on the Web.
- ***Relationship extraction*** is the extraction of relationships between named entities from the unstructured text or Web page.
- ***Sentiment Analysis or Opinion Mining*** is the technique to reliably extract reviews, opinions, and sentiments about various products and services from the content present on various online Web pages.
- ***Open/Surface Web*** is a small part of the Web that be easily accessed using popular search engines.
- ***Deep Web*** is the password-protected Web, or paid services, subscription services.
- ***Dark Web*** is the part of the Web that has hidden IP addresses and cannot be

directly accessed through popular search engines.

- ***Web search*** is searching for information from the Web**.**

CHAPTER 8

Social Network Analysis in Python

Introduction

With the evolution of Web 2.0, sharing of data on social networking sites has increased. It is not just about data; various information and services have also become online. People express their views on the internet using social networking sites, blogs, and various forums. In a way, we can see people are connected or related to each other. These relationships could be a one-way relationship (not a mutual relationship) or a bidirectional relationship (where both users are engaged in relationship). For example, Facebook follows the symmetric model, that is, both 'A' and 'B' are friends, whereas Twitter follows the concept of an asymmetric network (we will discuss in the next section), that is, if 'A' follows 'B', it will not mean that 'B' also follows 'A'. Learning about such relationships and analyzing them in a broader perspective can help answer many questions like who are the most influential people in the network? Or which set of people are more inter-connected (maybe from a particular group)? What are the types of relationship between various users in a group? For example, in a department, a common sight is that there are small groups that often go for lunch together. This demonstrates that they are more inter-connected; similarly, there are people who are part of almost all the groups, and such people are identified as the most influential people in the network. Knowing about such groups or people may help connecting across the network in minimum time for any activity. All these questions find their answer in social network analysis. Though social network analysis, in itself, is a big subject to study, but this chapter briefly introduces the readers to the concept of networks, social networks, different types of networks, how to analyze a network, and finally, a case study from Facebook and presentation of that data in meaningful knowledge.

Structure

Topics to be covered in this book are as follows:

- Introduction to social network analysis
- Creating a network
- Symmetric network
- Asymmetric network
- Network connectivity
- Network influencers

- Case study on the Facebook dataset

Objectives

The objective of this chapter is to introduce the reader to a very important area, which is social network analysis. The chapter will begin with an introduction to **Social Network Analysis** (**SNA**), creating a network using Python and various attributes related to SNA that helps to understand network behavior. We will learn and demonstrate how to find network connectivity and the role of network influencers in a network using python programs. At the end, we will discuss a case study on Facebook dataset to apply and understand the learnings of this chapter.

Introduction to Social Network Analysis

Historically the concept of "Social Networks" has been studied since the late 19th century and the beginning of the 20th century to explore the relationship between members of social systems. In looking back, academicians got fascinated with social media networks around the end of the 19th century when the scholars Georg Simmel, Emile Durkheim, and Max Weber studied about the structural perspective of the study of human behavior. However, the foundation of the study of modern social network analysis was laid by the psychiatrist Jacob Moreno. He studied about, how an individual's psychological well-being is linked to his or her relations with other individuals. Then after many lows and ups, moving through the work done by scholars of many universities like Harvard and the **University of California at Irvin** (**UCI**), who worked hard to popularize and promote social network analysis as a multidisciplinary area. Social network analysis is being used in many fields, such as psychology, sociology, economics, political science, sciences, and computer science also.

In recent years social network analysis has found many applications, such as finding the relationship between employees in an organization, Finding the person who can influence the most people in a network, contact tracing in a disease, and many others. We can understand **Social Networks Analysis** (**SNA**) as a methodical analysis of social networks.

A social network may be understood as a network of relationships between people and visualized as graphs, where nodes are people, and relationships are represented as edges.

In general, networks are considered as a set of objects (nodes) with interconnections (edges).

We have different types of networks as follows:

- Social networks
- Transportation and mobility network
- Information networks

- Biological networks
- Financial networks, co-authorship networks, and so on

But in this chapter, we will be focusing on social network analysis. In social networks, the emphasis is on the social structure between actors, mostly persons, groups, or organizations. It represents the way in which they relate to each other, and the relationship could be friendship, kinship, co-employees, and so on. So, we can say in social network analysis, we are doing an analysis of networks, wherein the network nodes are people or actors, and the edges are the relationship between people. **In SNA, the importance is on relationships not on people.**

Before going ahead, we must understand what is the need for doing this analysis. The aim of social network analysis is to understand a community by mapping the relationships that connect them as a network. It can also provide insights into social influences within the team and identify the problems. Following are some examples where social network analysis can be performed.

- Businesses use SNA to analyze and improve communication in the organization or with their partners or customers.
- SNA may also be used to identify criminal and terrorist networks from traces of communication or maybe identify key players in this network.
- Social networking sites (like Facebook) may use it for recommending potential friends based on friends-of-friends.
- To assess the performance of a student based on his/her social network, namely, classmates
- To analyze who is the most influential person in the network (organization)
- Looking at the network structure of a political party, we can predict if it is going to split into two structures and predict which members will go to which party or who is the most popular leader in the party
- Tracking movements
- Sentiment analysis

To understand, we can take an example of the relationship of students in a class. In a class, any two students may be connected to each other or not, the relationship between two students may be a one-way relationship, or it may be a two-way relationship (as shown in *figure 8.1*). Here, the direction of the arrow represents the relationship direction.

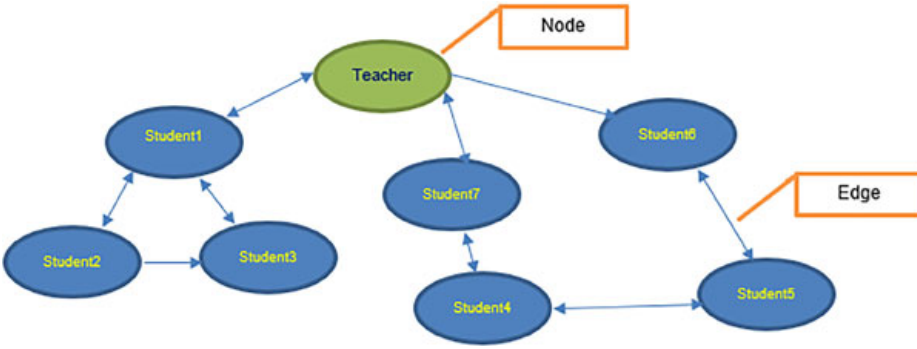


***Figure 8.1**: Relationship between one teacher and seven students*

In computer science, such relationships can be represented in terms of a Graph, also known as a Network.

A **network (or Graph)** is a representation of connections among a set of items. In such network items are represented as nodes (or vertices), and connections are called edges, as shown in *figure 8.2:*

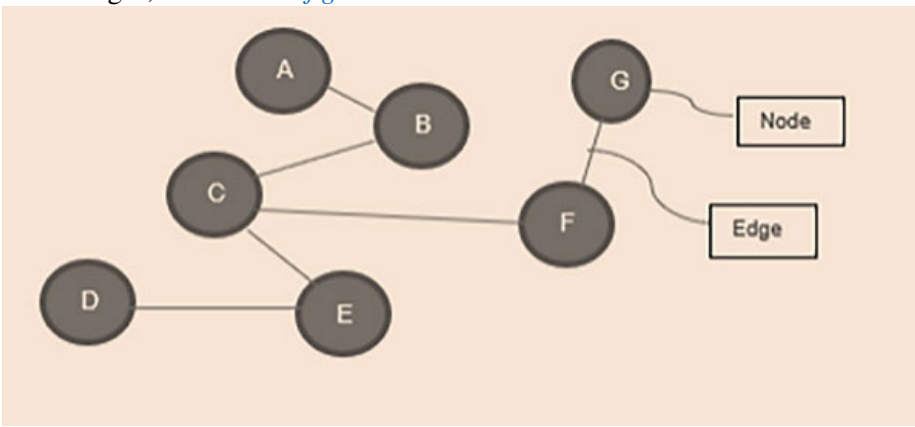


***Figure 8.2:** A network (graph) with nodes and edges*

Two important attributes to understand social networks are degree and density.

Degree: Degree refers to the number of connections a node has in a network. Such individuals with many connections can mobilize a large amount of resources and play a central role in the flow of information.

Density: Density refers to the ratio of the number of actual connections to all possible connections.

To demonstrate various concepts related to social network analysis, the Python programs given in this chapter are executed using Jupiter notebook on anaconda distribution. Code may be executed using Google Colab as well.

Creating a network

A network in Python is created using the library networkx. In order to create a network using Python, we need to import networks, then class Graph is used to create an object of graph type, and method add_edge() is used to add edges to the graph.

```
import networkx as nx

G = nx.Graph()      # Create a Graph object and assign it to G
## Further reference to graph will be through this variable

G.add_edge('A','B') # Add edge ('A','B') to graph G
G.add_edge('B','C') # Add edge ('B','C') to graph G
print(G.nodes()) # Print all nodes of graph G
print(G.edges()) # Print all edges of graph G
```

Output:

```
['A','B','C']
[('A','B'),('B','C')]
```

In this snippet, we have imported the library network as nx, added a graph as G and then added two edges (‘A’, ‘B’) and (‘B’, ‘C’); other edges can also be added in a similar way. Notice that we have not created any node, as the network automatically creates the connecting nodes if they are not existing when one creates the edges. The last two lines of the code display all edges and nodes of the graph created by the code.

> ***Note:***
>
> - You need to use “pip install networkx” if the library is not already installed.
> - Library network automatically creates the connecting nodes if they are not existing when one creates the edges.

In a similar manner, we can create a few more nodes, either by adding them separately or specifying all the nodes in the form of a list. See the following example:

```
import networkx as nx

G = nx.Graph() # Create a Graph object and assign it to G
## Further reference to graph will be through this variable

```

```
6.G.add_edges_from([('A','B'),('B',
('E','D'),('C','F'),('G','F')])
7.print(G.nodes())      Prinatl lnodeobs#f
graph
8.print(G.edges())      Prinatl ledgeobs#f
graph
9.
10.nx.draww(iGt,h_labs=eTlrsue#)# Printhseg raph
11.
```

Output:

```
['A', 'B', 'C', 'E', 'D', 'F', 'G']
[('A', 'B'), ('B', 'C'), ('C', 'E'), ('C', 'F'), ('E', 'D'), ('F', 'G')]
```

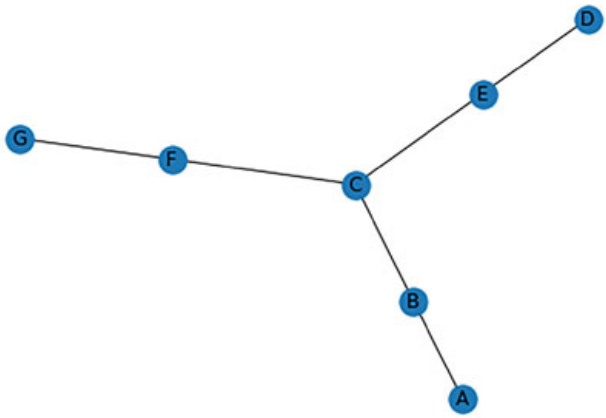


Figure 8.3: *A Network (graph) with nodes and edges*

Using the **a d d _ e d g e s _** method from Network X it is possible to pass a list of all edges to the method in order to add all edges of a graph in a single go instead of adding them separately.

Syntax: a d d _ e d g e s _ f r omf(e<dIgiesst>)

The method draw() draws the graph for visualization purposes. By default, it draws the graph without labels (*figure 8.4*). To display the labels of the nodes, one needs to pass the parameter "with_labels=True" (*refer to figure 8.3*).

Syntax: d r a w (G) # t dhregarwasp'hG 'w i t h o l a t b e l

Output:

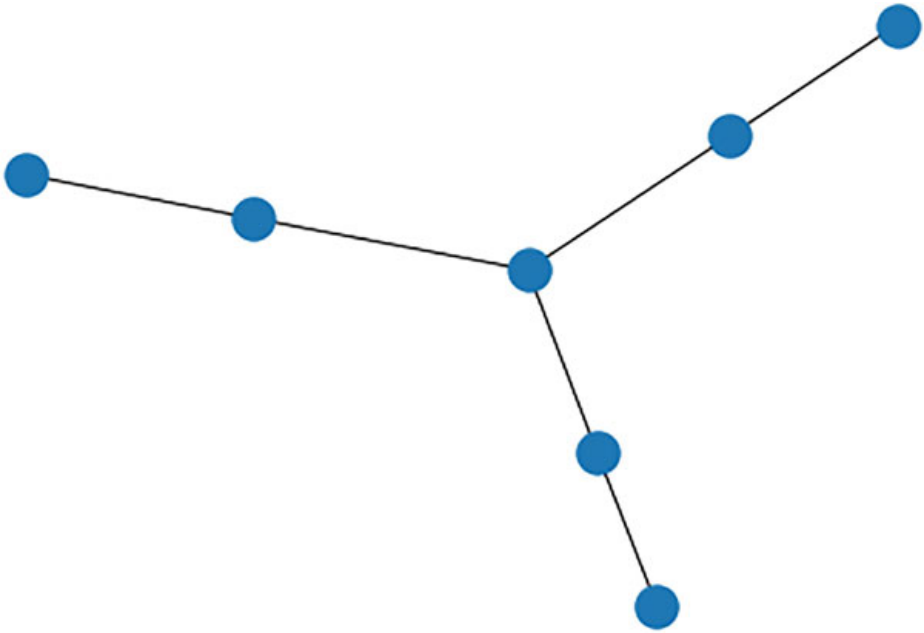


Figure 8.4: *A network (graph) with nodes and edges (without labels)*

Types of graphs

The networks have different attributes associated with them, such as direction, weight, and so on. Thus, based on these network attributes, the networks may be categorized as follow:

- Symmetric/undirected networks
- Asymmetric/directed networks
- Signed networks
- Weighted networks
- Multigraphs

Symmetric/undirected networks

Nodes and edges are building blocks of a graph and are used to represent a network, but how they represent the relationships can change our perspective of how we understand the network conceptually. Network that represents symmetric relationships are termed as Symmetric networks and are represented by undirected graphs. In a symmetric relationship, if 'A' is the friend of 'B', then 'B' is also a friend of 'A'.

Edges of the symmetric graph are undirected, and such graphs are created using the "Graph" class in Python. In such networks, the direction of edges does not have an arrow that represents the direction of the relationship. *Figure 8.5* represents one such network:

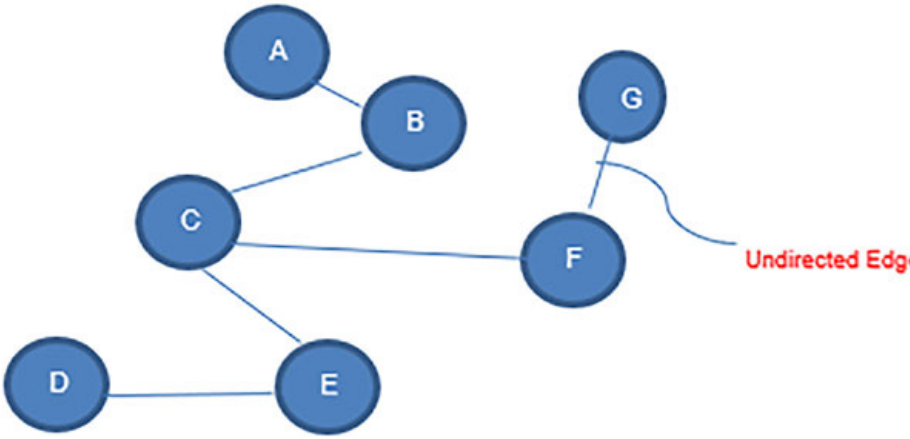


Figure 8.5: *Symmetric/undirected graph*

One of the examples of the symmetric network could be seen as a network of people who spend time together. When we say "Spending time together", it means both people are mutually involved in the relationship; it is not possible that person 'A' says that "I am spending time with 'B'" and 'B' says that "I am not spending time with 'A'". In terms of network theory, if the relationship between two people is "Symmetric" or "Undirected", it means there will be only a single edge between them; whether we call this edge 'AB' or 'BA' will not matter.

For example, Facebook follows a symmetric model that is both 'A' and 'B' are friends, whereas Twitter follows the concept of the asymmetric network (we will discuss in the next section), that is, if 'A' follows 'B', it will not mean that 'B' also follows 'A'. Some other examples of symmetric networks are "being in a same class", "being in the same club", and "being in the same seminar", which are also considered as symmetric networks as they all are co-members of the same group.

Symmetric network or undirected graph of the network presented in *figure 8.5* can be created using the following Python code:

```
1 .i m p o rnte t w o raksxn x
2 .
3 .G = n x . G r a p h# #C r e a taeG r a pchb j e catn da s s i g n
i t t oG
4 .# # F u r t hne f e r e nt oeg r a pwhi l eb e t h r o utgh i s
v a r i a b l e
5 .
6 .G . a d d _ e d g e s _ f r o m ( [ ( ' A ' , ' B ' ) , ( ' B ' ,
( ' E ' , ' D ' ) , ( ' C ' , ' F ' ) , ( ' G ' , ' F ' ) ] )
7 .p r i n t ( G . n o d e sP(r) i)natl #n#o d eosfg r a pGh
8 .p r i n t ( G . e d g e sP(r) i)natl #e#d g eosfg r a pGh
9 .
```

Output:

```
[ ' A '',B ' ; C ' ; E ' ; D ' ; F ' ; G ' ]
[ ( ' A " ,B ' ) (, ' B '',C ' ) (, ' C '',E ' ) (, ' C '',F ' ) (, ' E ' ,
' D ' ) (, ' F '',G ' ) ]
```

Asymmetric/directed networks

In contrast to symmetric relationships, there are some relationships that are not mutual. Such relationships in which only one actor is part of the relationship and other may not are represented by asymmetric networks. Asymmetric networks represent asymmetric relationships, that is, the relationships in which it is not necessary that if ‘A’ is a friend of ‘B’, then ‘B’ is also a friend of ‘A’. Such networks are represented by a directed graph where the direction of the arrow of the edge specifies the relationship between two nodes.

One very good example of an asymmetric network is the model used by Twitter. You can follow someone without them following you back, so this is a one-way relationship that may or may not be mutual (mutuality is not intrinsic to it). Unlike the model followed by Facebook, where if you accept someone as your friend, it means that the relationship is two-way or mutual. That is, everyone you claim to be your friend has also confirmed that you are also his friend.

One more good example could be modeling friendship between people. ‘A’ may consider ‘B’ as his friend, but ‘B’ may consider ‘C’ or ‘D’ or ‘E’ as a friend, but not ‘A’. This is an example of a one-way relationship, and such relationships can be modeled well using the directed graph; that is, such networks will also be known as symmetric networks.

Also, an excellent example of an asymmetric network is representing “Who eats who” in a network. For example, the relationship “eagle” eats “fish” or rabbit eats “grass” (in these relationships, vice versa is not true). Both these relationships are one-directional.

As discussed earlier, the edges of an asymmetric graph are directed, and such graphs are created using the “DiGraph” class in NetworkX. In such networks, the direction of edges matters, and thus, while creating the edges (‘A’, ‘B’) is different from (‘B’, ‘A’), that is, the order of writing ‘A’ and ‘B’ matters. In (‘A’, ‘B’), the direction of edge(relationship) is from ‘A’ to ‘B’ (meaning ‘A’ is related to ‘B’, whereas in (‘B’, ‘A’) the relationship is from ‘B’ to ‘A’ (*figure 8.6*).

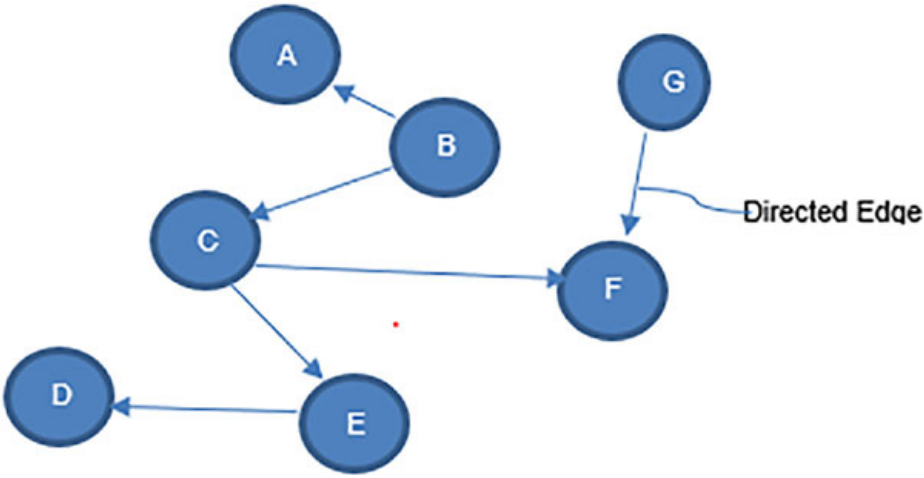


Figure 8.6: *Asymmetric/directed network/graph*

From the preceding graph, we may interpret the following relationships:

'B' is a friend of 'A', but 'A' is not a friend of 'B'

'B' is also a friend of 'C', but 'C' is not a friend of 'B'

'C' is a friend of 'F' and 'E', but the converse is not true

'G' is a friend of 'F', but 'F' is not a friend of 'G'

'E' is a friend of 'D', but 'D' is not a friend of 'E'

The following Python code will create the previously directed graph or Asymmetric Network:

```
import networkx as nx
G = nx.DiGraph()
G.add_edge('B','A')
G.add_edge('B','C')
G.add_edge('C','F')
G.add_edge('C','E')
G.add_edge('E','D')
G.add_edge('G','F')
nx.draw(G, with_labels=True)
```

Output:

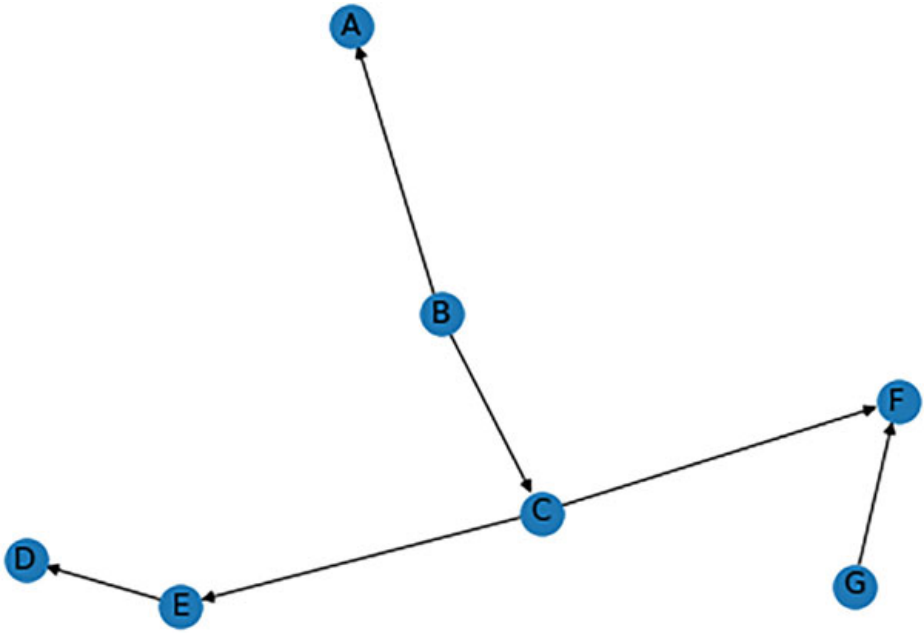


Figure 8.7: *Asymmetric/directed network/graph as generated from preceding code*

Note: In the case of an asymmetric network, the order in which nodes are written matters. While adding the edges to the network, ('A', 'B') is NOT the same as ('B', 'A').

Signed networks

The concept of assigning "sign" or "weight" to these networks is also being used to make these networks more informative and meaningful.

For example, in a symmetric network, the mutual relationship could be "positive" or "negative". That is, the two actors may be positively connected to each other (friends) or negatively connected to each other (foes). The networks in which the relationship (symmetric or asymmetric) is represented using signs ("+" or "–") are termed as **Signed Network**.

Such networks often reflect a mixture of positive (friendly) and negative (antagonistic) interactions:

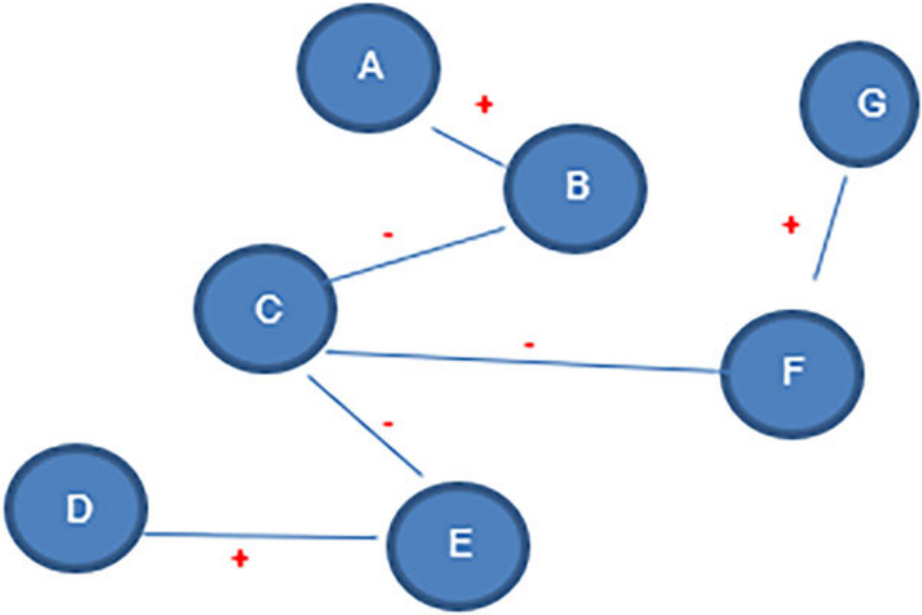


Figure 8.8: *Signed network/graph*

From this network diagram (*refer to figure 8.8*), we can infer that nodes 'A' and 'B' are friends, whereas nodes 'B' and 'C' are having a negative relationship.

Weighted networks

Similarly, in a network, any relationship, it is difficult that all the relationships are equally important. Such relationships are represented by assigning weight to the relationships. In the graph, the weight is written on the edge. In weighted networks, the edges/relationships are assigned different weights, mostly numerical, that represent the strength of that relationship. For example, if we have a network representing the number of times co-workers have been for coffee together, in such networks, the nodes will be the co-workers, and the edges will carry a weight that will represent the number of times they have gone for lunch together (*see figure 8.9*).

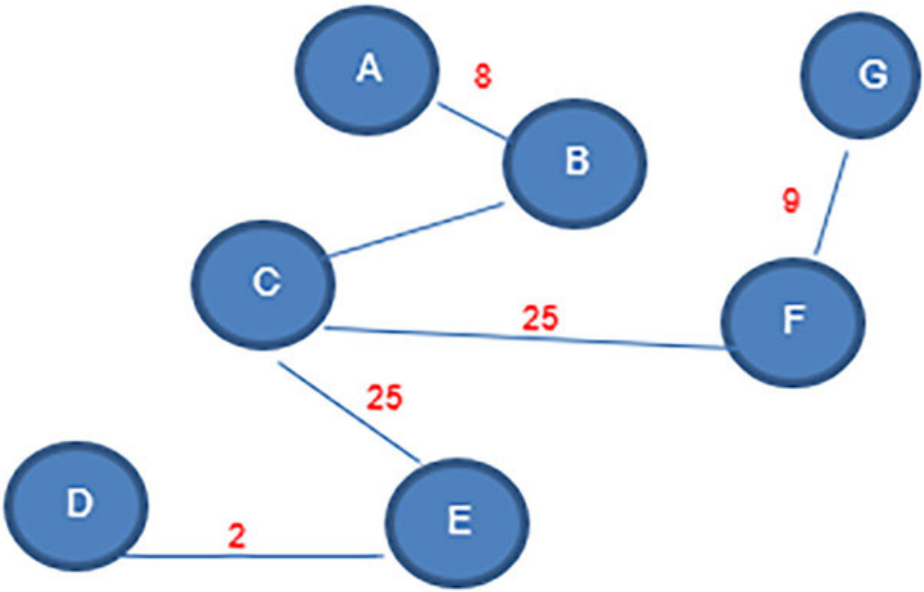


Figure 8.9: *Weighted network/graph*

We can interpret from this graph that 'A' and 'B' together have gone for coffee eight times, whereas 'C' and 'F' have gone for coffee 25 times. Such networks are also known as **Weighted Networks**. Another good example of the weighted network can be representing the case in which we want to model the number of times each employee has sent an e-mail to another.

Some more attributes may be used to depict relationships between two actors and written on the edges. For example, we can write labels on the edges to depict the type of relationship between two nodes as 'Family', 'Friend', 'Extended Family', and so on. Such relationships can be represented while creating network using Python as follows (*figure 8.10*):

```
1 . i m p o rnte t w o raksxn x
2 . G = n x . G r a p h ( )
3 . G . a d d _ e d g e ( ' Ae ł a Bi ǫ n = ' F a m i l y ' )
4 . G . a d d _ e d g e ( ' Be ł a Ci ǫ n = ' F r i e n d ' )
5 . G . a d d _ e d g e ( ' Ce ł a Fi ǫ n = ' E xFtaemnidleyd' )
6 . n x . d r a ww(iGt,h _ l a b=eTlrsu e )
7 .
```

Output:

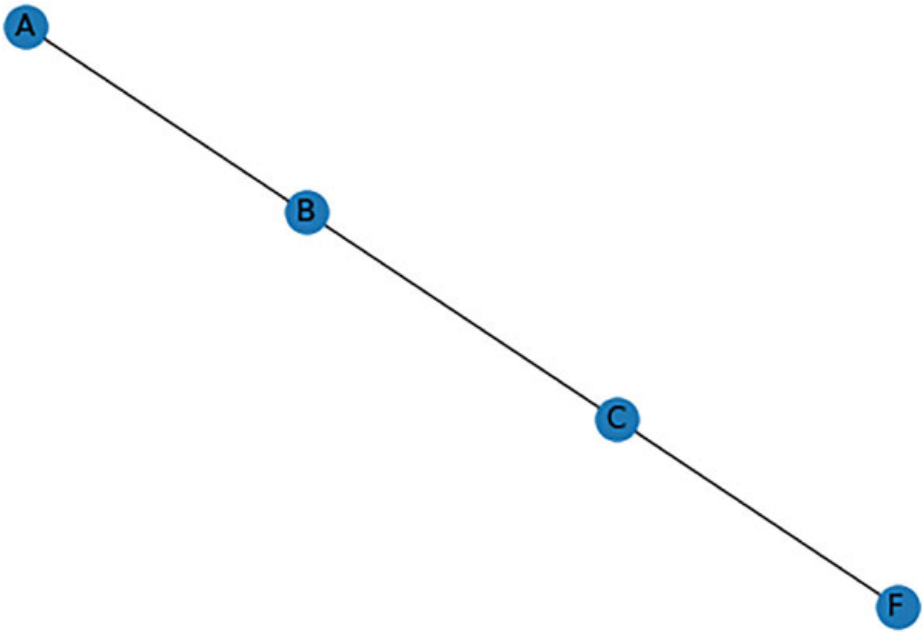


Figure 8.10: *Graph having labeled edges (relations)*

Multigraphs

We may have networks where in a graph, the two nodes may have multiple relationships simultaneously. For example, 'A' and 'B' may be 'Friend' as well as a member of 'Extended Family'. **Such graphs, in which the two nodes have multiple relationships, are known as Multigraphs** and can be created using `MultiGraph()` class in Python (*see figure 8.11*). In such graphs, the two nodes will have multiple edges representing multiple relationships, where each edge represents one relationship.

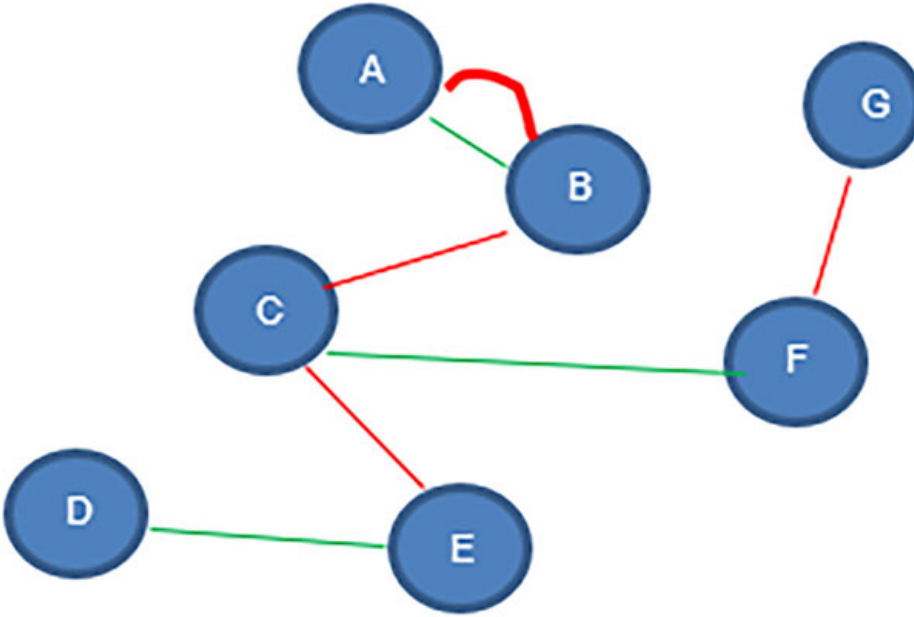


Figure 8.11: *Multigraph network/graph*

In this network, green lines represent 'Friend', and 'Red' lines represent 'Extended Family'.

Analyzing network

After having seen all the different types of networks and various attributes that can be represented on the edges of the network, we now see various Python methods that help us to understand the nature of the network. Python method **edges()** is used to display all the edges of the related graph. For further details with respect to attributes associated with the edges, we can use the same method **edges()** with the parameter "**data = True**".

```
1.import networkx
2.G=nx.Graph()
3.G.add_edge('A','B',relation='Family')
4.G.add_edge('B','C',relation='Friend')
5.G.add_edge('C','F',relation='Extended Friend')
6.nx.draw(G,with_labels=True)
7.
```

In order to list all edges, we use **G.edges()** as follows:

```
1.print(G.edges())
```

Output:

```
[('A','B'),('B','C'),('C','F')]
```

In order to list all edges along with their attributes, we use G.edges() with

parameter data-True as follows:

```
1.print(G.edges(data=True))
```

Output:

```
[('A','B',{'relationship':'family'}),('B','C',
{'relationship':'friend'}),('C','F',{'relation':
'Extended family'})]
```

In order to list all edges having a particular attribute, we use G.edges() as follows:

```
1.print(G.edges(data='weight'))
```

Output:

```
[('A','B',8)]
```

It is also possible to access the attribute of a specific edge ('A', 'B') in this case, as follows:

```
1.G.edge(['A']['B'])
```

Output:

```
['Relation':'Family','weight':8]
```

We can see it returns a dictionary showing each attribute with its value associated with this edge.

It is also possible to access a specific attribute associated with a particular edge as follows:

```
1.G.edge(['A']['B']['weight'])
```

Output:

```
8
```

These methods will apply to all types of networks that we have discussed so far, that is, symmetric network, asymmetric network, signed network, weighted graph, and multigraph network.

Note: As this is the undirected graph, the order of nodes does not matter, so if we write ['A']['B'] or ['B']['A'], it does not. But in the case of a directed graph, we will have to take care of the order else you will get an error.

Just like the edges can have attributes, even the nodes can have attributes. For example, actors in relationship "number of time had coffee with" could have some designation associated with them; for example, 'A' may be "Director", 'B' may be a "Coordinator" and 'C' may be a "Computer Operator". These can also be specified while creating the network using NetworkX.

```
1.importnetworkasnx
2.G=nx.Graph()
3.G.add_edge('A','B',weight=6,family
4.G.add_edge('B','C',weight=3,frien
```

```
5.G.add_node('A',role='Director')
6.G.add_node('B',role='Coordinator'
7.G.add_node('C',role='Computer Operator')
8.print(G.nodes())
9.print(G.nodes(data='True'))
10.
11.nx.draw(G,with_labels=True)
```

To access all the nodes, we can use the method **nodes()** as follows:

```
1.G.nodes()
```

Output:

```
['A','B','C']
```

Similarly, G.nodes(data=True) will display all nodes with the attributes as follows:

```
1.G.nodes(data='True')
```

Output:

```
[('A',{'role':'Director'}),
{'role':'Coordinator'},{'role':'Computer
Operator'})]
```

Also, if we want to access the role of a particular node, instead of nodes(), we use node() and specify the name of the node and attribute as follows:

```
1.G.node['A']['role']
```

Output:

```
'Director'
```

Distance measures in network connectivity

The value of any social network is determined by its size and the connectivity between the nodes. The connectivity of a network is used to quantify various attributes of a network, such as the shortest path, nodes that are farthest (yet connected), the nodes that are closest, the node through which other nodes are connected, and so on. The nodes may be connected directly or indirectly. Some of important terms that help to understand network connectivity in a network are discussed in the following paragraphs:

Distance

There are many situations where we need to know how far a node is from another? or maybe we would like to know if some nodes are close to each other and others are far or are they equidistant? And then, we may like to know which nodes are close to each other and which nodes are far from each other?

To answer all these questions, we have an attribute that is known as "**Distance**" *that is used to measure the shortest distance between any two nodes.*

The distance can be computed based on the "path" that is chosen to traverse from one node to other. For example, the path between node "K" to "I" can be K-B-C-E-I, or it can be K-B-C-F-E-I (*see figure 8.12*).

We first create the network (as shown in *figure 8.12*) to understand how a network can be analyzed.

Thus, the distance between node 'K' to 'I' can be computed based on both these paths and is defined as the shortest path length between two nodes.

In Path 1, that is, K-B-C-E-I the path length can be computed as the number of edges that needs to be traversed to reach node 'I'. So, the distance between 'K' and 'I' is 4 as the edges that are traversed to reach node 'I' are 'K-B', 'B-C', 'C-E', and 'E-I'.

Similarly, in Path 2, that is, K-B-C-F-E-I the path length will be 5 as the edges traversed are 'K-B', 'B-C', 'C-F', 'F-E', and 'E-I'.

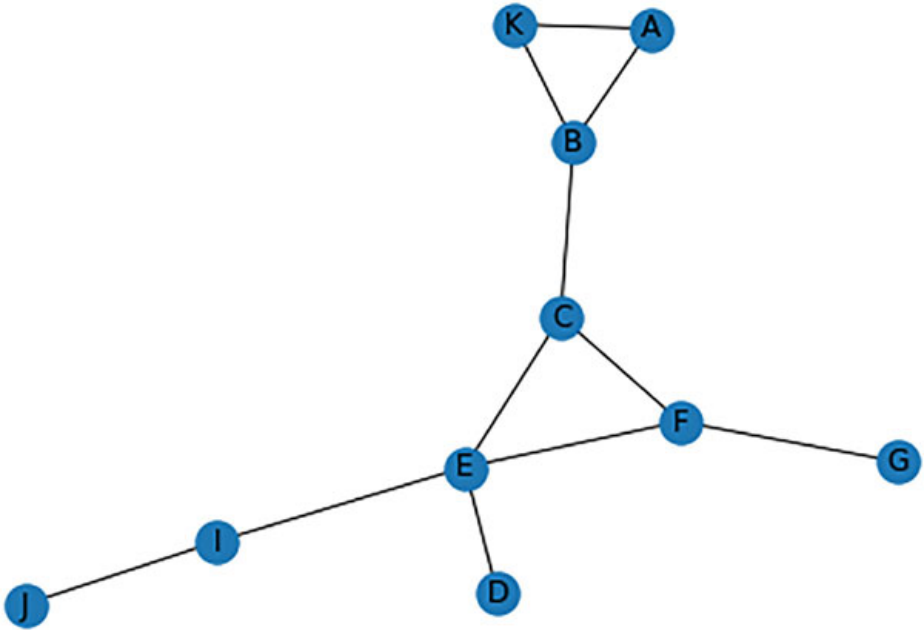


Figure 8.12: *Random network/graph*

So, the distance between these two nodes is 4, which is the shortest path length between the said nodes. If we have a larger network with a large number of nodes, this may not be a suitable method to find out the path length and the distance. Networkx provides a method shortest_path() and shortest_path_length() to find the shortest path and distance between any two nodes, respectively. Both these functions require graph names and the name of the nodes between which the distance is to be computed.

```
1.nx.shortest_path(G,'K''I')
```

Output:

```
['K', 'B', 'C', 'E', 'I'}   path//shorte
```

Now, we compute the shortest length as follows:

```
1.nx.shortest_path_length(G,'K''I')
```

Output:

```
4        //shortpeastthlengthhafis,distance
```

To find the distance from one node, say 'A' to all other nodes, Networkx uses the breadth-first search algorithm by using the function **bfs_tree()**. We need to pass the graph name and the node from where the distance is to be computed.

```
1.T=nx.bfs_tree(G,'A')
```

The function **bfs_tree()** returns the tree of the graph, where we can find all the edges using function **T.edges()** and the distances of all nodes from 'A' using **shortest_path_length()**

```
1.T.edges()
```

Output:

```
[('A','K"A,',','B')B',','C('')C,',','E('')C,',','F'}
('E','D('')E,',','I('')F,',','G('')I,',','J')]
```

We can see **T.edges()** returns a list of all the edges in the tree 'T'. Now we compute the distance of all nodes from a particular node, which in this case is 'A' as follows:

```
nx.shortest_path_'Aength(G,
```

Output:

```
{'A'0:,
'B'1,
'C'2,
'D'4,
'E'3,
'F'3,
'G'4,
'I'4,
'J'5,
'K'1}
```

You may see that the result is stored in the form of key-value pairs in a dictionary, where "key" if the name of the node and "value" is the distance between node 'A' and the given node stored in "key". We can interpret this as 'A' has a distance "0" from itself, a distance "1" from 'B', and so on.

Note: Breadth-first search (BFS) algorithm is used to traverse a tree data structure. It starts at the root node and searches for all the nodes at the present depth before moving to the nodes of the next level (depth).

Average distance

The average distance between all the nodes is defined as the number of steps along the shortest path for all possible pairs of network nodes and can be computed by the function **average_shortest_path_length()** provided by NetworkX.

```
1.nx.average_shortest_path_length(G)
```

Output:

```
2.5111111111111111
```

Eccentricity

Eccentricity is used to find out the largest distance from a node to other nodes in a graph. Method **eccentricity()** provided by NetworkX to find eccentricity and need graph name as a parameter. We define "eccentricity" as the largest distance between a node and all other nodes and compute it as follows:

```
1.nx.eccentricity(G)
```

Output:

```
{'A':5,
'B':4,
'C':3,
'D':4,
'E':3,
'F':3,
'G':4,
'I':4,
'J':5,
'K':5}
```

We can understand from this result that the eccentricity of 'C' is 3, which means no node has a distance of more than 3 from node 'C', and node 'K' has an eccentricity of 5, which means the maximum distance between node 'K' and the farthest node is 5.

Diameter

The Diameter represents the maximum distance between any two nodes in the network. It is computed by the function **diameter()** which requires the graph name as a parameter.

```
1.nx.diameter(G)
```

Output:

```
5
```

Thus, the Diameter is the maximum distance between any two nodes in the

network, that is no node will have distance more than 5. **Diameter is equal to max eccentricity among all the calculated eccentricities.**

Radius

Radius is defined as the smallest distance between any two nodes. So, from the preceding example, we can see that eccentricity of 'E' is 3, which represents the minimum distance between any two nodes. So, we can say that **radius is the minimum eccentricity.** radius() method of NetworkX calculates radius of the graph.

As of now, we have a clear idea about distance in a network, and we have seen that the largest distance is calculated as the diameter of the network, and the small distance is computed as radius. Now, we would like to find which nodes are far away from the other nodes and which nodes are closer to other nodes. Periphery gives us an answer to our first question, that is, which nodes are far away from other nodes?

```
1 .n x . r a d i u s ( G )
```

Output:

```
3
```

Periphery

We can define **the Periphery** of a graph as the set of all nodes that have eccentricity equal to the diameter. In this example, we can see that nodes 'A', 'K', and 'J' have eccentricity equal to the diameter, that is, 5. NetworkX function **p e r i p h e r y ()** can be used to compute for larger networks, (*refer to figure 8.13*).

```
1 .n x . p e r i p h e r y ( G )
```

Output:

```
[ ' A ' ',K ' ; J ' ]
```

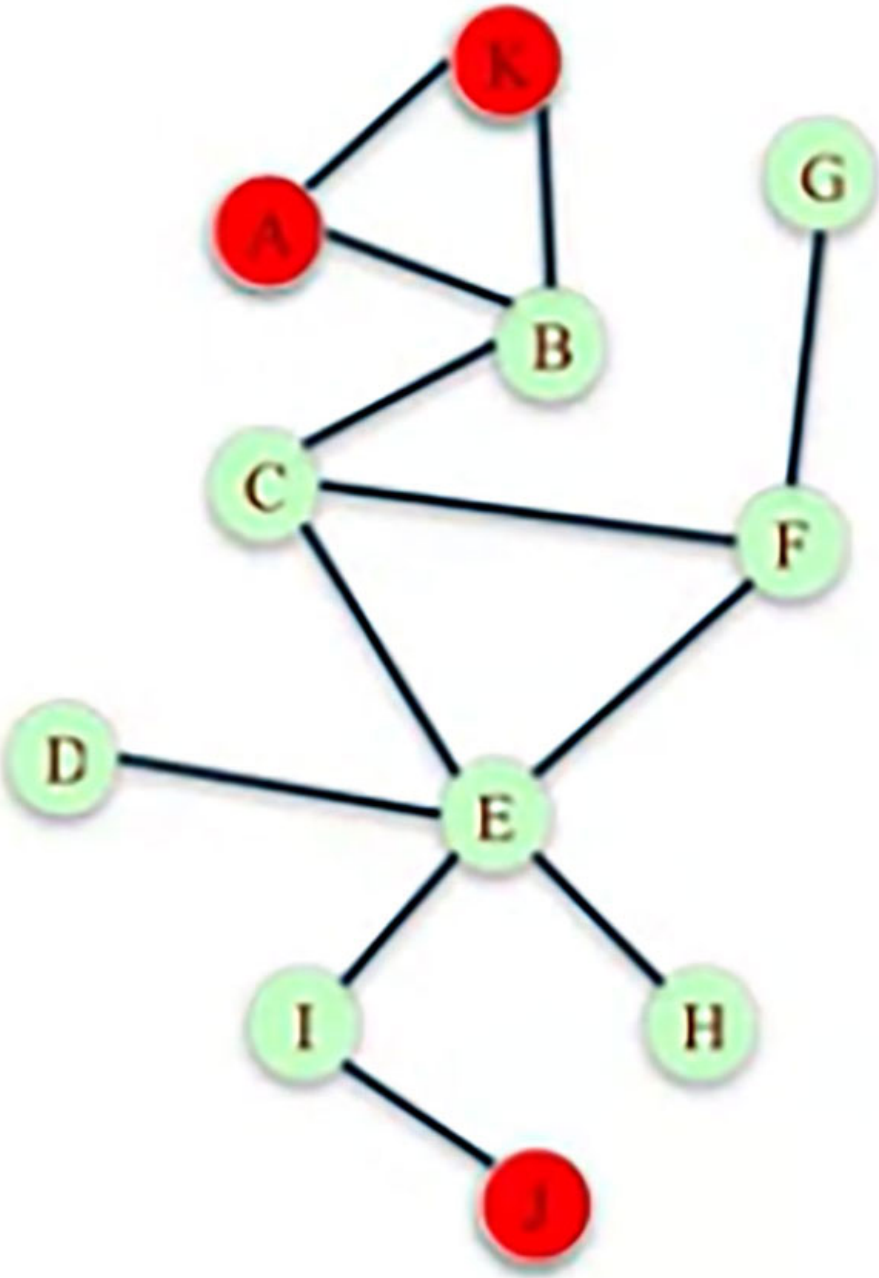


Figure 8.13: *Peripheral Nodes 'A', 'K', and 'J'*

Center

Another important attribute related to distance measures is the **center**. "*The center may be defined as set of nodes whose eccentricity is equal to radius of the graph.*

So, the nodes that are at the center are close to all other nodes" (refer to figure 8.14).

```
1 .n x . c e n t e r ( G )
```

Output:

```
[ ' C ' ',E ' ; F ' ]
```

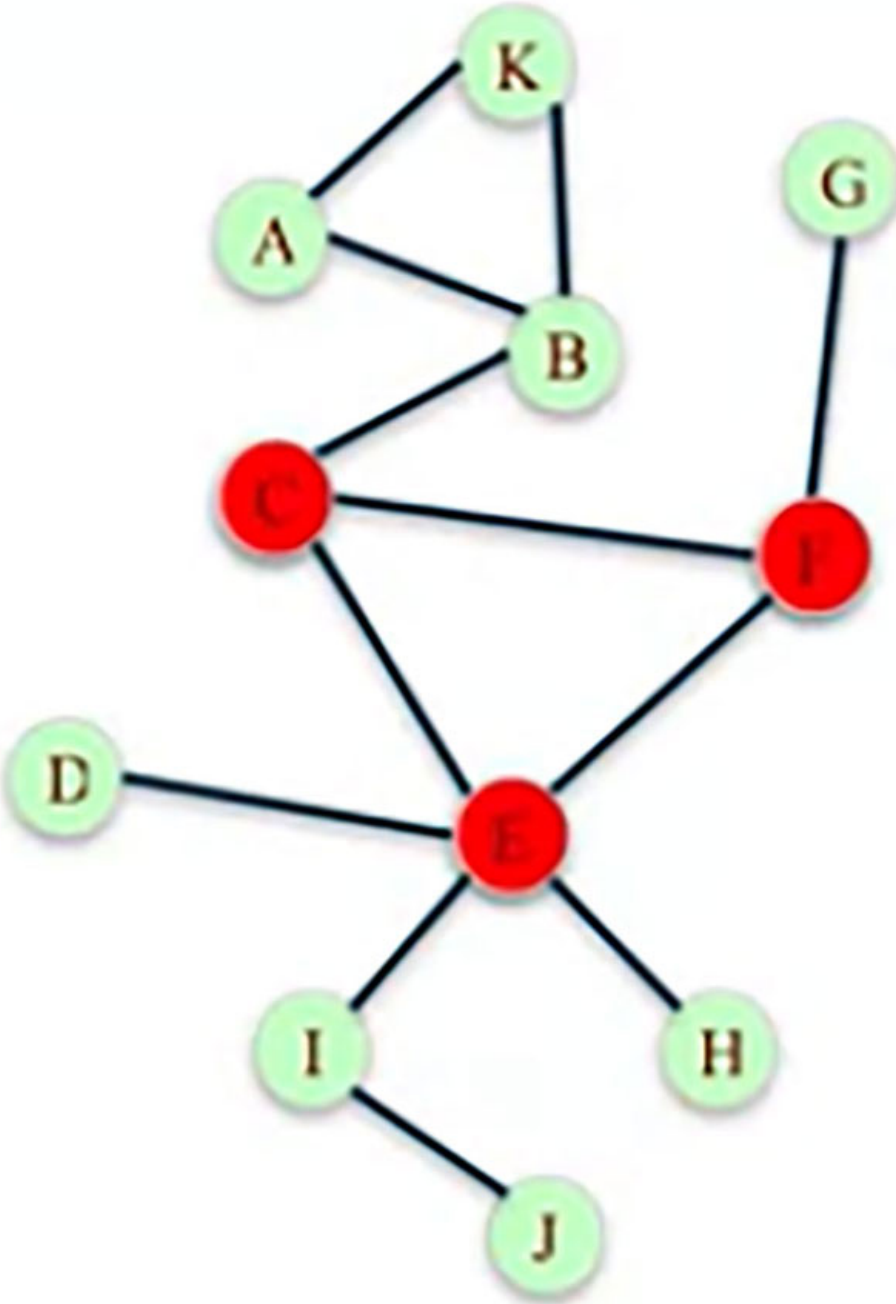


Figure 8.14: *Graph representing central nodes 'C', 'E', and 'F'*

So, we can infer that nodes 'C', 'F', and 'E' are close to all other nodes.

Network influencers

As per the report on global social media statistics published by "datareportal", there will be 4.70 billion social media users by July 2022. In these networks, there are people having thousands and millions of followers, and thus, are seen as people who can influence people who are following them, both positively and negatively. Followers look up to these people to guide them in taking their decision. Such people who have built up a reputation for their knowledge and expertise in a specific domain and can influence the decision-making of people who follow them are known as **Network Influencer.** A network influencer has access to a large audience in their network, and by virtue of their authenticity and reach, they can persuade this audience. Owing to this property, network influencers can help to have better control over the spread of rumors, conduct successful advertisements for e-commerce products and optimize resources in digital marketing.

Wide use of various social networks has greatly enriched our life. One of the most fascinating branches that are emerging from the use of social networks is the use of related social media and network influencers' to promote brand awareness and market products and services. This could include paid advertisements or designing viral content. Based on the benefits that can be reaped by the popularity and followers of network influencers, it is worthwhile to identify such influencers. Graph theory is a widely used model for social networks; thus, in this section, we will explore how to find network influencers in a social network.

To find how important are the nodes in a graph, we need to study about the centrality measures. Centrality measure is vital tools for understanding networks and gives us information on how important nodes are in a graph. Based on the information related to how many connections it has and what other nodes is it connected to, they can have a wide range of influences on a graph.

Popular centrality measures are **Degree Centrality, Closeness Centrality, Between Centrality**, and **Eigenvector centrality**.

Degree centrality measures the number of edges that are connected to a node and is used to find the node that is most connected. To find the normalized degree centrality of a node, add the number of edges that are attached to the node, divided by the total number of nodes minus one. NetworkX provides a function `degree_centrality()` to find the centrality measure. The node with the largest centrality is the node with the maximum connections.

Closeness centrality is a measure that measures the mean distance from one node to any other node. The more central a node is, the closure it is to all nodes. To find the closeness, NetworkX provides a function called `closeness_centrality()`. The node with the highest closeness centrality is said to be closest to the most nodes than all other nodes.

Between centrality measures the number of shortest paths that the node lies on and is thus used to determine the flow of information through the graph. The higher the number, the more information flows through it. For the between-

centrality measure, NetworkX provides the function `betweenness_centrality()`. The highest betweenness centrality means that it lies on most of the shortest paths and thus maximizes the information flow.

Eigenvector Centrality is the measure of node's relative influence on the network, that is, how well a node is connected to other highly connected nodes. Calculating this centrality measure is comparatively more difficult than other centrality measures, but the network makes it easier. Function `eigenvector_centrality()` provided by the network that computes this centrality measure. The node with the highest eigenvector centrality is the node which can influence maximum nodes.

Social network analysis can be used in varied applications, as mentioned at the beginning of the chapter. The following section presents a use case study of social network analysis where we can see how all these measures and attributes can help us analyze any network.

Note: In directed graphs, degree centrality is separate for incoming edges (known as in-degree) and outgoing edges (known as out-degree).

Case study on Facebook dataset

In this section, we will discuss a case study that will help us to understand the concepts that we have learnt so for. Data for the study may be obtained using Web mining; however, in our example, we are using the Facebook dataset that is made available for study purposes by Stanford university. The dataset contains details of various connections on the Facebook page and is used to analyze the relationships of various connections. Nodes are various connections, and edges are the relationships.

This dataset consists of "circles" (or "friends lists") from Facebook. Facebook data was collected from survey participants using this . The dataset includes Edges from 10 egonets combined.

Note:

- Egonet is some node's network that includes its friends and friendships between them. Egonet may also be understood as a subgraph that includes all of the central node's friends and all edges between them.
- Facebook dataset can be downloaded from: **https://snap.stanford.edu/data/ego-Facebook.html**.

Through our example, we will try to answer various questions using the measures that we have defined earlier in this section:

1. What is the shortest path between any two nodes (Distance)
2. What is the average distance between various nodes in a network (Average

Distance)

3. What is the maximum distance from a node to any other node (Eccentricity)
4. Which are the two nodes that are farthest yet connected (Diameter)
5. Which are the two nodes that are closest to each other (Radius)
6. Which nodes are the centre nodes (through which all nodes are connected) of the network? (Center)
7. Which are the most influential nodes? (Centrality measures)

In order to perform an analysis of the graph, we will first read the dataset using the **read_edgelist** method. We provide the filename along with the location where it is stored (Dataset/facebook_combined.txt) as an argument to this function. Method info() displays the common statistics about the graph, such as the number of nodes, edges, and average degree.

```
import networkx
G = nx.read_edgelist("Dataset/facebook_combined.txt")
print(nx.info(G))
```

Output:

```
Name:
TypeGraph
Number of nodes4039
Number of edges88234
Average degree:  43.6910
```

We can see the number of nodes as 4039, the number of edges as 88234, and the average degree as 43.6910:

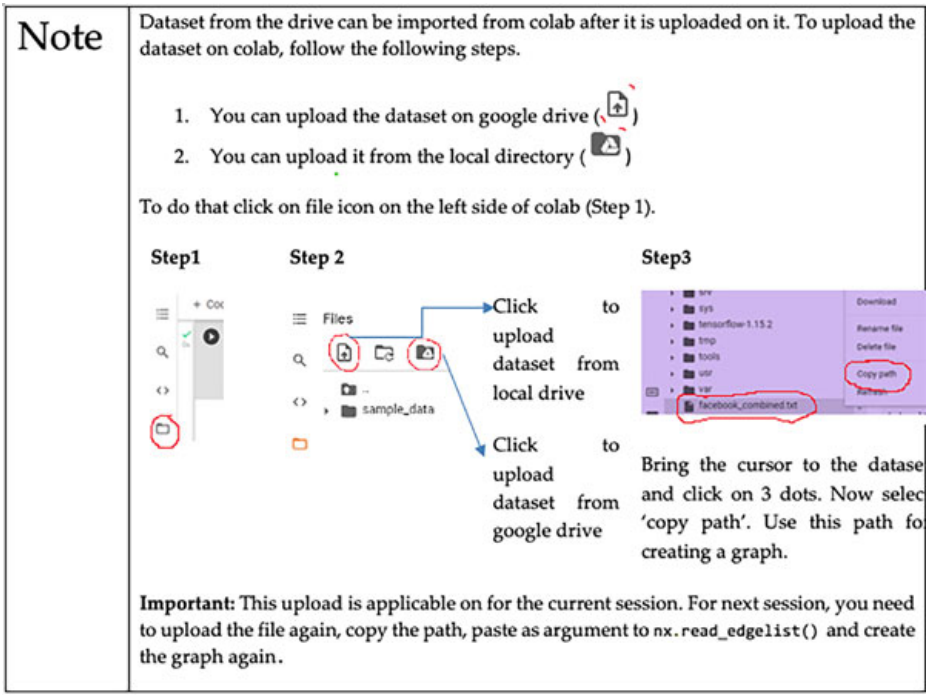


Figure 8.15: *Figure to demonstrate uploading of dataset on Google Drive*

We now find all nodes with the shortest path using the `shortest_pa` function shown as follows. The following output shows the various shortest path that exists in the network.

And the average distance between the nodes is computed using `average_shortest_path`. From the result we can see that the average distance between the nodes comes out to be `3.6925068496963913`

```
1.nx.average_shortest_path_length(G
```

Output:

```
3.6925068496963913
```

The maximum distance from a node to any other node (eccentricity) is computed as shown:

```
1.nx.eccentricity(G)
2.
```

Output:

```
{'0':6,
'1':7,
'2':7,
'3':7,
'4':7,
'5':7,
```

```
' 6 ' 7 ,
' 7 ' 7 ,
' 8 ' 7 ,
' 9 ' 7 ,
' 1 0 '7:,
' 1 1 '7:,
' 1 2 '7:,
' 1 3 '7:,
' 1 4 '7:,
' 1 5 '7:,
' 1 6 '7:,
' 1 7 '7:,
' 1 8 '7:,
' 1 9 '7:,
' 2 0 '7:,
…
' 1 4 9 9 6 ,:
' 1 5 0 0 6 ,:
' 1 5 0 1 6 ,:
' 1 5 0 2 6 ,:
' 1 5 0 3 6 ,:
' 1 5 0 4 6 ,:
' 1 5 0 5 5 ,:
' 1 5 0 6 6 ,:
' 1 5 0 7 6 ,:
' 1 5 0 8 6 ,:
' 1 5 0 9 6 ,:
' 1 5 1 0 6 ,:
' 1 5 1 1 5 ,:
' 1 5 1 2 6 ,:
' 1 5 1 3 6 ,:
… }
```

Looking at the output, we see that the distances are 5, 6, and 7. As per the definition, the maximum distance of one node from other node is eccentricity. So, for this network, the eccentricity is 7.

The diameter of the network tells us how big the network is, and the radius of the network is used to find the closest nodes. The following code computes both the diameter and radius, as we see that the diameter of the network is 8, that is, the maximum distance between any two nodes in the network is 8, that is, you can reach to the farthest point of the network by 8 hops. The minimum distance that is radius is 4, which is the minimum.

```
1 . p r i n t ( " D i acmfegtreapGhi s :" , n x . d i a m e t e r ( G
```

Output:

```
Diametoefgraph is :    8
```

Similarly, the radius can be computed as follows:

```
1.print("Raodfigursaph is:",nx.radius(G))
```

Output:

```
Radiwsfgraph is :    4
```

Then comes the next important attribute, that is, the center node, which represents the nodes through which all nodes of the network are connected. If we remove this node, the network will break. `center()` returns a list of all such nodes, and from our case study, we can see there is just one central node, that is node '567' in this dataset, usually it is a list of nodes.

```
1.print("Ceonftgeraph is:",nx.center(G))
```

Output:

```
Centeorfgraph is:['567']
```

> ***Note:*** We can also find other things like the peripheral nodes, connected nodes and so on. I leave this as an exercise to the reader of the book.

Now, we will compute centrality measures to find out the influential nodes. We can use the degree centrality measure to find the number of nodes attached to a node closeness centrality measure to measure the mean distance from one node to any other node. The more central a node is, the closer it is to all the other nodes; the between centrality measure is used to measure the number of shortest paths that the node lies on. This centrality is usually used to determine the flow of information through the graph. The higher the number, the more information flows through it, and the eigenvector centrality measures the node's relative influence in the network or how well a node is connected to other highly connected nodes. The more the value, the more influential the node is. The following code and out output show the degree and the eigenvalue centrality of the network.

```
1.print("Nordeel'astiivnefluemnteh enetwork
graph is\n",x.eigenvector_centrality(
```

Output:

```
Node'rselatiivnefluemnteh enetwogrkaph is:
{'0':3.391796172270200 5e-05,
'1':6.045346134948106e-07,
'2':2.233460937191196 3e-07,
'3':6.635648083921 05e-07,
'4':2.236415702889359 8e-07,
'5':1.183322111843521 2e-06,
'6':2.170904169516125 5e-07,
'7':2.568510124822488 e-05,
'8':2.197374633483027 7e-07,
```

```
'9'2.2141166244860936e-06,
'10'7:.7085670440926 2e-07,
…
'40303.:3111997242431 82e-10,
'40322.:9497312752978847e-10,
'40332.:9507415709655 47e-10,
'40342.:9512702918327 91e-10,
'40352.:9129014284709 78e-10,
'40362.:9312234293826837e-10,
'40372.:9892325138276 5e-
```

Similarly, we also find the degree centrality measure of the graph to find the number of nodes attached to a node.

```
1.print("Numofnodeattachtedanodeina
networkgraphG",nx.degree_centrality(G))
```

Output:

```
Numberofnodeattachtedanodeinanetworkgraph
G:
{'0'0:.085933630510153 54,
'1'0.004210004952947003,
'2'0.002476473501733531 3,
'3'0.004210004952947003,
'4'0.002476473501733531 3,
'5'0.003219415552253590 7,
'6'0.001485884101040118 8,
'7'0.004952947003467063,
'8'0.001981178801386825,
'9'0.014115898959881128,
'10'0:.002476473501733531 3,
…
'107'0:.258791480931154
…
'4035'0.:0002476473501733531 3,
'4036'0.:0004952947003467063,
'4037'0.:0009905894006934125,
'4038'0.:0022288261515}
```

We can see the largest degree is of node "**0.2587914809311154**" which may be considered as the node through which a maximum number of nodes are connected. However, the eigenvalue of node '1509' is the largest, that is, 8.98; thus, this is the most influential node. Other nodes with a higher value for this centrality measure are also nodes with greater influence.

Though a lot more can be studied about the SNAs, their attributes, and many other things related to social network analysis (which itself is a subject), it is beyond the

scope of this book.

Conclusion

This chapter discussed the concepts of Social Network Analysis and its applications in various day-to-day works. We also learned about the type of networks, important attributes that help us understand the network, various parameters to find the level of connectivity in a network and discussion about network influencers and their importance in a network.

In the end, we concluded with a case study of the Facebook dataset in which we tried to answer some questions based on the dataset.

In the next chapter, we will discuss about "Web usage mining". Web usage mining is to discover interesting usage patterns from Web data using data mining techniques. Usage data captures the origin of Web users along with the browsing history and behavior on a website.

Points to remember

- Historically the concept of "Social Networks" has been studied since the late 19th century and beginning of the 20th century to explore the relationship between members of social systems
- A social network is a network of relationships between people and visualized as graphs, where nodes are people and relationships are represented as edges.
- SNA is used to analyze relationships in a network.
- Concept of graph is used to represent social networks where the item is represented as nodes (or vertices), and connections are called edges.
- **Degree**: Degree refers to the number of connections a node has in a network. Such individuals with many connections can mobilize a large amount of resources and play a central role in the flow of information.
- **Density**: Density refers to the ratio of the number of actual connections to all possible connections
- Python library NetworkX is used to create and analyze these networks.
- Symmetric/undirected networks are used to represent symmetric relationships (where relationship is mutual)—for example, Facebook
- Asymmetric/directed networks are used to represent Asymmetric relationships (where relationship is not mutual)—for example, Twitter. Direction of the relationship is shown using the head of the arrow.
- Not all relationships are equal. Weighted networks represent the weight of relationship on edges.
- Multigraphs are used to represent the nodes that can have multiple relationship.

- "Distance" is used to measure the shortest distance between any two nodes.
- Average distance between all the nodes is defined as the number of steps along the shortest path for all possible pairs of network nodes.
- Eccentricity is used to find out the largest distance from a node to other nodes in a graph.
- Diameter represents the maximum distance between any two nodes in the network
- Radius is defined as the smallest distance between any two nodes.
- The periphery of a graph as the set of all nodes that have eccentricity equal to diameter.
- Center of a graph is the set of nodes that have eccentricity equal to the radius
- Centrality measures are vital tools for understanding networks.
- Popular centrality measures are Degree Centrality, Closeness Centrality, Between Centrality and Eigenvector centrality.
- Degree centrality measures the number of edges that are connected to a node and is used to find the node that is most connected.
- Closeness centrality is a measure that measures the mean distance from one node to any other node.
- Between centrality measures the number of shortest paths that the node lies on and is thus used to determine the flow of information through the graph.
- Eigenvector Centrality is the measure of node's relative influence on the network, that is how well a node is connected to other highly connected nodes.

Multiple choice questions

1. Which of the following best explains social network analysis?

    a. Platform to send e-mails

    b. Stores network configuration details

    c. Made up of social interactions/connections between people.

    d. None of the above

2. It is graphical representation that consists of Nodes and _____.

    a. People

    b. Networks

    c. Computers

    d. Edges

3. Directed networks are also known as _____.

a. Symmetrical Network
b. Asymmetrical Network
c. Weighted Network
d. Signed Network

4. Which of the following are an example of a social network?

a. Mobility Network
b. Co-authorship networks
c. Employee in an Organization
d. All of the above

5. Maximum eccentricity is same as _____ in a network.

a. Radius
b. Average Distance
c. Distance
d. Diameter

6. Minimum eccentricity is same as _____ in a network.

a. Radius
b. Average Distance
c. Distance
d. Diameter

7. Which measure is used to find the node that is close to all nodes?

a. Centre
b. Radius
c. eccentricity
d. None of these

8. In a telecommunication network, which measure will be used to find node having more control over network (through which maximum information will pass through that node).

a. Between Centrality
b. eccentricity
c. Degree Centrality
d. None of the above

9. In a network, the edges may contain several information. Can nodes also contain any information?

a. True

b. False

10. Which nodes are the nodes through which all nodes are connected in a network?

    a. Radius
    b. eccentricity
    c. Centre
    d. None of these

11. _____ is the measure of node's relative influence on the network, that is, how well a node is connected to other highly connected nodes?

    a. Between Centrality
    b. Eigenvector Centrality
    c. Degree Centrality
    d. None of the above

Answers

Questions

1. Define Social Network Analysis.
2. What are the various applications of Social Network Analysis?
3. What are the different types of networks in terms of social network analysis? Give examples to explain each.
4. Differentiate between directed networks and undirected networks. Give example to illustrate.
5. Network connectivity can be understood with different attributes of the network. Mention five and explain their relevance.
6. Who are network influencers? How to find the most influential nodes?

Key terms

- **Social Network Analysis** (**SNA**)
- Directed/ Undirected graph
- Symmetric/ Asymmetric Network
- Signed Networks
- Weighted Networks
- Multigraphs
- Distance

- Average Distance
- Diameter
- Radius
- Eccentricity
- Periphery
- Degree
- Center
- Centrality Measures
- Degree Centrality
- Closeness Centrality
- Between Centrality
- Eigen vector centrality

CHAPTER 9

Web Usage Mining

Introduction

We have learnt that the Web has enormous storage of Web pages and links. When user access and search the Web for any information, the user's access is traced and stored in access logs.

Nowadays, even for any basic information, we search the internet. This dependency has resulted in the requirement for an increase in data, and near about one million pages are added every day. The log files are created when a user/ customer interacts with a Web page. Now since this need for information search has resulted in the growth of Web pages, which, in turn, increases the Web log file. Web usage mining relates mining techniques in log data to extract the usage patterns from Web data. Web usage mining helps to understand the need of users/ customers and serve the needs of Web-based applications in a better way.

Structure

In this chapter, we will discuss the following topics:

- Sources of data
- Types of data
- Key elements of Web usage data pre-processing
- Data modeling
- Discovery and analysis of pattern
- Predictions on transaction pattern

Objectives

Web usage mining is to discover interesting usage patterns from Web data using data mining techniques. Usage data captures the origin of Web users along with the browsing history and behavior on a website. This data is analyzed to discover patterns and predict user behavior. This chapter introduces the reader to the concepts of Web usage mining, key elements of Web usage data pre-processing, modeling, discovery, and analysis of patterns and predictions based on it.

Process of Web usage mining

Before we start the process of mining, it is very much required that the Web log

data is pre-processed and pattern analysis is performed. Pre-processing of data is essential because the Web log data is noisy as well as confusing. There might be a question in your mind, that is, why Web log data is confusing? So, the answer is Web log data is confusing because it contains irrelevant and unambiguous data. It is very much required to pre-process this data before applying any Web mining algorithm; otherwise, it may affect the result of the mining process.

The process of Web usage mining may be categorized into the following three stages:

1. Data collection and pre-processing
2. Pattern discovery
3. Pattern analysis.

In the pre-processing stage, the cleaning of clickstream data is performed. The clickstream data is the information collected about a user while they browse through a website or use a Web browser. A clickstream is the record of an individual's clicks during Web surfing. The Click Stream data contains the basic information about visited pages, and along with this, it may also include information like how much time a user spent on a given page, what page features they engaged in, and where they went next.

Furthermore, the data is partitioned and presented into a set of user operations. These sets of operations or transactions basically represent the activities of individual users while they visit the site. The user transaction data may be enhanced by using semantic domain knowledge, site structure, and so on. The semantic domain knowledge can be used from site ontologies.

The prime objective of the second phase, that is, pattern discovery is to obtain the hidden pattern from the navigation/Web surfing performed by the user. Machine Learning and Statistical techniques are performed so that the characteristics of user behavior can be reflected and detailed statistics on parameters such as sessions, Web resources, and so on can be presented.

One could ask the question, what is the use of a collection of clickstream data and then pre-processing and pattern discovery? So, the answer is that pre-processed data and patterns lead to the formulation of models, which are basically used as a primary input for many applications such as Web analytics, recommendations systems, and so on. To achieve this comprehensive model, the outcome of the second stage is further processed and filtered in the final stage, that is, the pattern analysis stage. *Figure 9.1* depicts the various stages of Web usage mining.

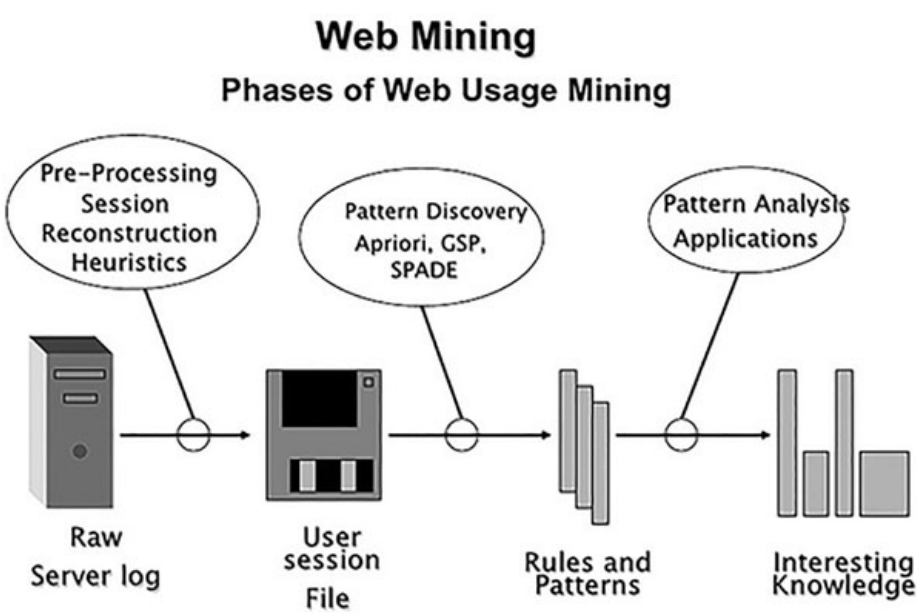


Figure 9.1: *Stages of Web usage mining [Google]*

Sources of data

The server log files, including Web server access logs and application server logs, are the main data sources used in Web usage mining. The site files and meta-data, operational databases, application templates, and domain knowledge are additional data sources that are crucial for both data preparation and pattern identification.

Since companies have realized that data is a rich source of information, they are focusing on using specialized services from Data Aggregation Companies. Data Aggregators are companies that facilitate data exchange by connecting a consumer or business' financial accounts to authorized fintech partners, providing information the companies need to power their services. This is very key for the growth of any enterprise. Data aggregation helps summarize data from different, disparate, and multiple sources. It increases the value of information. The best data integration platforms can track the origin of the data and establish an audit trail. We can trace back to where the data was aggregated from. This information is typically unavailable, and if it is, it is typically not kept in a single, centralized location. The enterprise is responsible for locating the data sources it requires to get insights. It is also responsible for aggregating that data for analytics.

A few top Data Aggregation Companies are as follows:

- MX
- Finicity
- Fiserv/CashEdge
- Mobius Services

- Yodlee
- 3iDataScraping
- Plaid and Quovo

Types of data

Data collected across multiple sources are basically classified into four primary groups. *Figure 9.2* depicts the classification of data. Each category is discussed in detail in the next section:

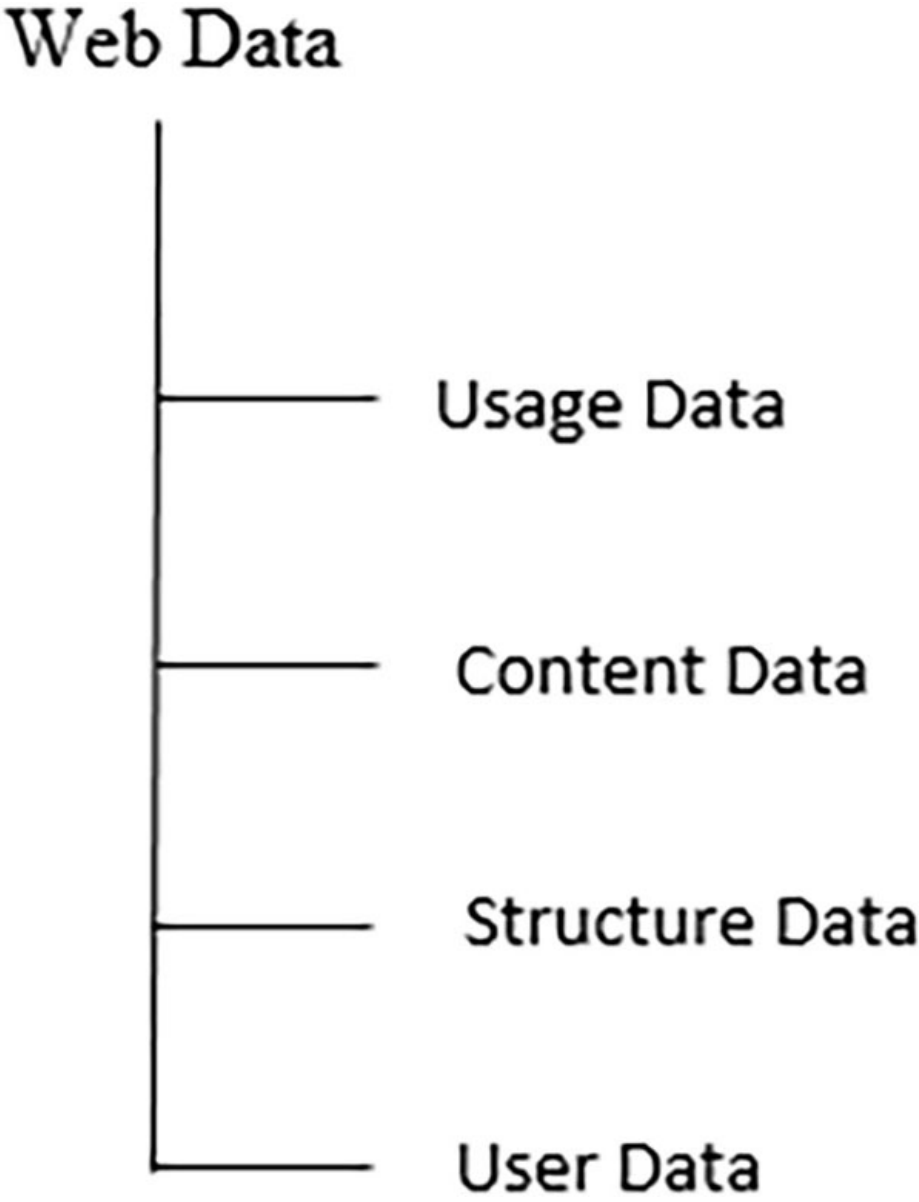


Figure 9.2: *Classification of Web data*

Usage data

The Web and application servers automatically collect log data, which is a representation of the visitors' precise navigational patterns. In Web usage mining, it serves as the main source of data. One record is created in the server access logs for each hit against the server, which corresponds to an HTTP request. Depending on the log format, each entry in the log might include the following information:

- Fields having the time and date of the request
- Client's IP address
- Detail of the requested resource
- Any potential parameters used to invoke a Web application
- Request's status
- The HTTP method used
- The user agent—version and type of browser and operating. The Web resource being referred
- To report the repeat visitor, client-side cookies may be used.

Data may be aggregated at different levels of abstraction. The level of abstraction depends upon the objective of the analysis. If we talk about Web usage mining, pageview provides the utmost basic-level abstraction. It is basically an accumulated presentation of all those objects which contribute so that with a single click, the required information is displayed on the client's screen.

Google defines a pageview as "An instance of a page being loaded (or reloaded) in a browser". Every time someone starts a session by opening and viewing a page on your website (regardless of whether they are loading or reloading it), you get one pageview.

If we talk about the user level, then the session is the most basic level of abstraction. A session is basically a series of pageviews by a single user for the duration of a single visit.

Google Analytics defines a session as "A session is a period of activity completed by a user. Google Analytics records a session every time someone visits your website. Sessions start as soon as a page is opened, and cookies are activated, and end after users have been inactive for 30 minutes by default."

Google Analytics 4 provides a complete view of consumer behavior across the Web and app by using first-party, modeled data. The improved machine learning features, actionable reporting, and new integrations offer reliable insights with evolving privacy and technology.

Google Analytics toolkit may help you to track pageviews. In *figure 9.3*, a sample of pageview from the Google analytics toolkit is shown (page names are hidden for

a purpose).

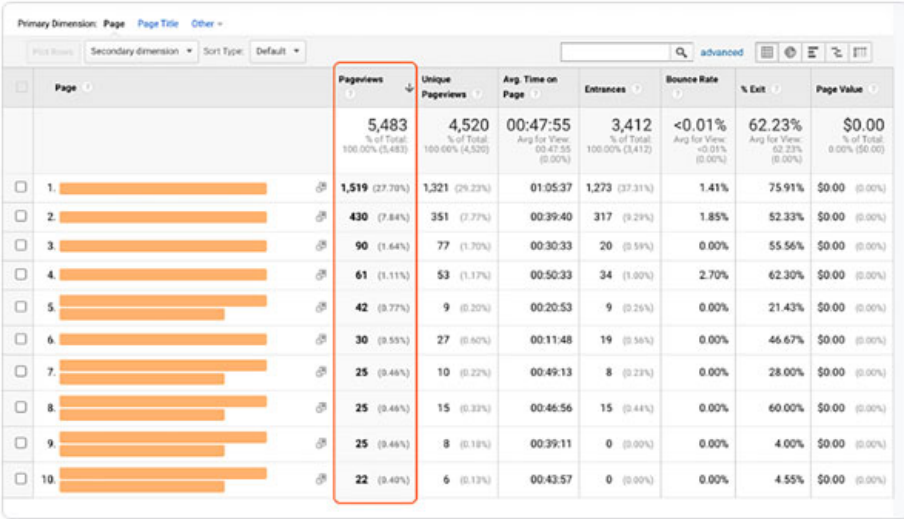


Figure 9.3: *Pageview from Google Analytics toolkit*

Note: The number of pageviews does not represent the number of users. The reason is that one user may contribute multiple pageviews in one session by refreshing the page.

Content data

The collection of objects and relationships that are communicated to the user through a website constitutes the content data. This data is primarily made up of mixtures of textual and visual components. Static HTML/XML pages, multimedia files, dynamically created page segments from scripts, and collections of records from operational databases are some of the data sources utilized to transmit or generate this data. Along with descriptive keywords, document characteristics, semantic tags, or HTTP variables, the site content data also comprises semantic or structural meta-data that is incorporated into the site or pages.

Structure data

The structural data reflects the site's content organization, as seen by the designer. This organization is represented by the hyperlinked inter-page linking structure between pages.

The intra-page organization of the material on a page is also included in the structure data. For instance, across the space of tags in the page, both HTML and XML texts can be seen as tree topologies.

User data

Additional user profile information may be present in the operational database(s) for the website. Such data may include demographic details about registered users, user reviews of various items, such as books or movies, user visits, purchase histories, and other overt or covert indications of users' preferences. If it is possible to tell which users are which, some of this data can be collected anonymously.

Key elements of Web usage data pre-processing

Data pre-processing consists of data cleaning, user identification, session identification, path completion, and transaction identification. *Figure 9.4* represents various stages of Web usage mining.

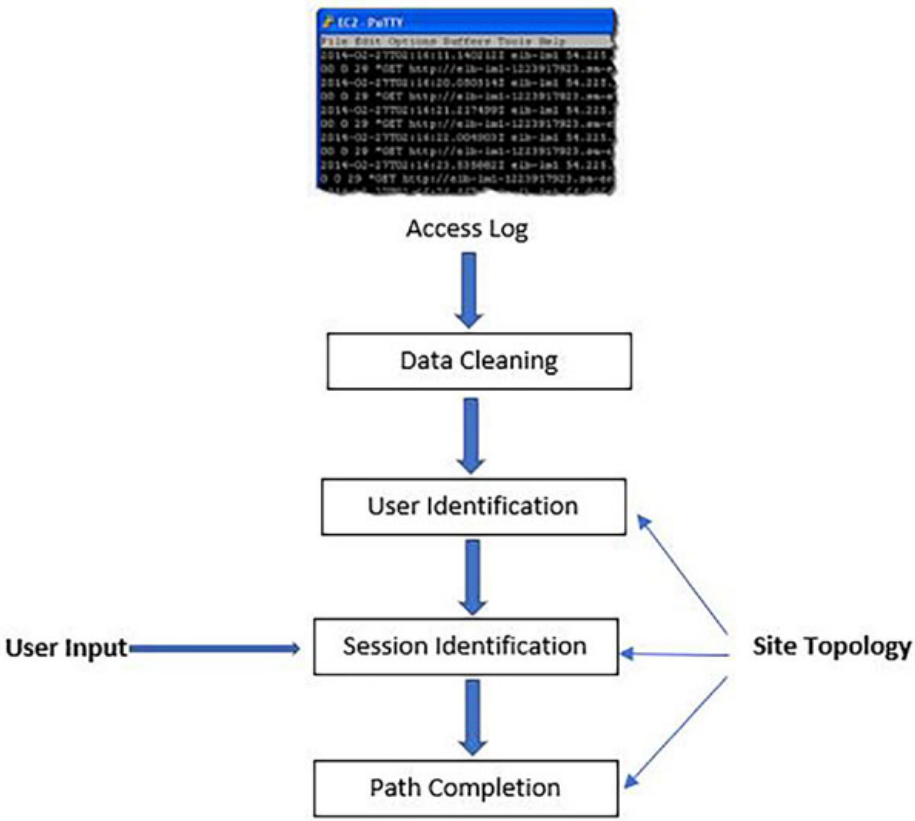


Figure 9.4: *Stages in Web usage mining*

Data cleaning

Data cleaning typically involves actions that are site-specific, such as deleting superfluous references to embedded objects, such as style, graphics, or sound file references, that may not be necessary for the analysis. Due to crawler navigation, references are also removed at the data cleaning phase.

It reduces the size of log files. It is essential for improving the effectiveness of the

mining process. It consists of the following phases:

- removal of local and global noise
- removal of records of graphics, videos, and format information
- removal of records with the failed HTTP status code
- method field and robots cleaning

User identification

Individual users accessing a particular website could be identified through the user identification process. The basic objective of this phase is to offer personalized services to the end user. The objective is achieved by extracting the characteristics of a specific user from the cluster of users. On the internet, all the users may be identified by their unique IP address. In other words, we can say that every IP address relates to one specific user. But there do exist a few possibilities as follows:

- Few users may have a unique IP address
- Few users may have two or more IP addresses
- And few users may share one IP address.

For the identification of users, the following rules are followed:

- If there are different IP addresses, then it will be considered as a different user.
- If the IP address is the same, but the browser is different, or the operating system is different, then, in both cases user must be considered as different.
- Now, if the IP address, browser, and operating system are all the same, then the user identification will do by checking whether the requesting page could be attained via pages that were accessed before. This identification is done according to the site topology.

Session identification

The practice of segmenting each user's user activity record into sessions, each of which represents a single visit to the site, is known as session identification or sessionization. Web log mining typically spans a wide time frame, so users may visit the site more than once.

The purpose of session identification is to separate the access records into different accessing sequences, each of which involves simultaneously requesting several pages.

It determines how many times a user has accessed a certain Web page, compiles all the page references for that user in a log, and then divides them into user sessions.

In classification, prediction, clustering, and other mining functionalities, these

sessions can be used as data vectors. It is considered a fresh user session if the URL in the referrer URL field of the current record has never been viewed previously or if it is empty. It can be difficult to reconstruct precise user sessions from server access records. The standard timeout is the foundation of the session identification technique. A new session is established when the time between two consecutive requests exceeds the timeout. Websites that lack features like embedded session ids and additional user authentication data must rely on heuristic methods for sessionization or session identification. A sessionization heuristic's objective is to recreate the real series of activities taken by one user during one visit to the website using clickstream data.

Path identification

Path completion is a pre-processing activity that might be crucial and is often done after sessionization. Discovering the existence of significant accesses that are not listed in the access log is crucial. The insertion of crucial page accesses that are omitted from the access log by caching proxy and browser servers is known as path completion. Missing access references to those pages or objects that have been cached are frequently a consequence of client- or proxy-side caching. For instance, if a user visits page A again while still in the same session, it is likely that the second access to A will display the previously downloaded copy of A that was cached on the client side, so no request is sent to the server. As a result, the server logs do not contain the second reference to A. Path completion, which is based on the understanding of site structure and referrer data from server logs, can be used to heuristically infer missing references because of caching. The referrer log can be used to determine from which page the request originated, like how user identification works when determining whether a page a user requests is directly related to the one they previously accessed. It is assumed that the user browsed back using the "back" button and cached sessions of the pages if the page is in their recent click history. The pages which are backtracked are also included in the complete path reflected by a session.

Data modeling

Web servers can store a lot of information in logs from user interactions on Web pages. Geographical data, their journey around the website's pages, and many other things are included. Four different types of data mining techniques are typically used to harvest user-generated data from the Web. Let us go over each of these methods in greater detail:

Association rule mining

One of the fundamental Data Mining techniques that developers typically employ for Web usage mining is the association rule. Using this technique, the website can monitor user information and deliver recommendations based on their browsing

habits and search history. The Association Rule's fundamental precept is divided into two parts: an Antecedent (if condition) and a Consequent (then condition). An item found in data is called an antecedent, while an item found together with an antecedent is called a consequential. Consider a customer looking for protein powder in an online store (Antecedent). Different protein powders, mass gainers, protein shakers, and other products could be the Consequent for this Antecedent product. In Web usage mining, association rules are used to discover connections between pages that commonly follow one another in user sessions.

Sequential pattern

In a significant amount of sequential data, the sequence is found using sequential patterns. In Web usage mining, sequential patterns are used to identify users' navigational patterns. Because the sequential patterns develop gradually over time, they serve as a definition for the order of occurrences. For the creation of sequential patterns, there are two categories of algorithms.

Based on association rules mining, the first algorithm to recognize sequential patterns was developed. For example, two well-known Apriori algorithms for extracting association rules are GSP and AprioriAll.

The second approach for finding sequential patterns represents them using Markov chains and tree structures. For instance, one of these algorithms, WAP-mine, uses the WAP-tree tree structure to investigate Web access trends.

Clustering

Among a large amount of data, clustering is a technique used to group comparable items together. The distance function, which determines the degree of similarity between various elements, can be used for this grouping. In Web usage mining, clustering is a technique for gathering comparable experiences.

There are the following two distinct clustering approaches available:

- User clustering
- Pages clustering

Classification mining

Classification mining is focused on creating a profile of things that fit into a certain group and are categorized in accordance with their similar characteristics. Any newly added things can be classified by the profile, and the database can be updated accordingly.

Based on the available demographic data, classified mining in Web mining enables the developers to create a profile for users who view specific goods.

Discovery and analysis of pattern

In order to identify usage patterns that indicate user behavior and interests, significant correlations between data are mined from the available usage data. To automate the pattern discovery process, learning techniques, including clustering, association rule discovery, and sequential pattern discovery, are frequently used at this step.

The following step in the **Web Utilization Miner** (**WUM**) process comprises the finding of usage patterns that can be correctly exploited to realize the functionality of the Web application once the Web usage data has been pre-processed. This goal is pursued by using a variety of techniques and algorithms from several research areas, including statistics, data mining, machine learning, and pattern recognition, which can extract meaningful information about user browsing patterns from usage data.

By using conventional statistical methods on session data, most commercial programs regularly mine knowledge about user activity. As a result, several WUM traffic tools offer periodic reports that summarize crucial statistical data describing user browsing patterns, such as the most frequently visited pages, the average amount of time spent viewing a page, the average length of navigational pathways, and so on. The collected information can be used to accomplish a variety of objectives, including enhancing system performance, making site change tasks easier, supporting marketing decisions, and so on.

Web usage statistics are provided by many commercial software. The usage data offered by traditional website trackers may be sufficient for tiny Web servers to study usage patterns and trends. The statistics supplied by current Web log file analysis technologies may prove insufficient, and more sophisticated knowledge-mining approaches will be required as the bulk and complexity of the data expands.

This chapter focused on four algorithms: association rules, sequential patterns, clustering, and classification in the context of knowledge discovery approaches specifically for Web usage data.

Association rule for knowledge discovery

The simplest method for finding patterns in WUM is association rule mining, which basically looks for meaningful connections between collections of pages that regularly appear in user sessions.

The creation of association rules can be used to connect pages that are frequently referred to one another inside a single server session. Association rules in the context of Web usage mining refer to groups of pages that are accessed collectively with a support value greater than a predetermined threshold. There may not be a direct hyperlink connecting these pages to one another. For instance, association rule discovery using the Apriori algorithm may show a correlation between people who accessed a page about athletic equipment and those who visited a page with electronic products. The existence or lack of such criteria might aid Web designers in reorganizing their website. These guidelines are applicable not only for

commercial and marketing applications but also for other purposes. The association rules may also be used as a heuristic for prefetching documents to lessen how long it takes for a user to detect a page loading from a distant site.

Pattern discovery through clustering

The most popular method used in the process of finding patterns is clustering. As discussed previously, there are basically two objectives to perform clustering in Web usage mining.

First, one is finding user groups with similar browsing habits is the basic goal of user clustering. With the help of such knowledge, user demographics can be used to perform market segmentation in E-commerce applications or provide personalized Web content to the users.

Second, the goal of Web page/document clustering is to identify collections of pages with similar content. Web documents have additional attributes like metadata or hyperlinks that can be used to divide them apart in addition to their content. Link-based document clustering techniques use data gleaned from the link structure of sites to pinpoint collections of related pages.

Sequential pattern mining for knowledge discovery

The identification of navigational patterns in Web usage data proves to be particularly helpful for sequential pattern discovery. This method involves adding time to the pattern-finding process. This method aims to identify time-ordered collections of pages that pop up frequently during user sessions. Web marketers can place adverts targeted at particular user groups by using this method to forecast future visit trends. Sequential patterns can also be the subject of trend analysis, change point detection, or similarity analysis, among other types of temporal analysis.

Learning through classification

Classification has been used in the specific context of WUM to describe user behavior and the attributes of websites to establish their membership to a given category. In general, the extraction and selection of features that may adequately characterize the characteristics of the provided classes come first in the classification process.

The classification of things into predetermined classes is then performed using supervised learning methods. Using learning strategies such as decision trees, naive Bayesian classifiers, and neural networks can be accomplished.

For example, classification on server logs may lead to the discovery of interesting rules, such as 40% of users who placed an online order in /Product/Sports are in the 16–20 age group and live in New Delhi.

Pattern analysis

Filtering out uninteresting rules or patterns from the collection discovered during the pattern discovery phase is the driving force behind pattern analysis. The application for which Web mining is being done typically dictates the precise analysis methodology. Pattern analysis is most frequently performed using a knowledge query engine like SQL. Another way to carry out OLAP operations is to put usage data into a data cube. Techniques for visualizing data, such as charting patterns or assigning colors to various values, can frequently draw attention to broad patterns or trends in the data. Filtering out patterns that comprise pages with a specific usage kind, content type, or pages that match a specific hyperlink structure can be done using content and structure information.

Note: Online Analytical Processing (OLAP) is a category of software that allows users to analyze information from multiple database systems at the same time. OLAP business intelligence queries often support in trends analysis, financial reporting, sales forecasting, budgeting, and other planning purposes. It is a technology that enables analysts to extract and view business data from different points of view.

Predictions on transaction pattern

The main objective of a generic WUM process is to mine usage patterns that simulate Web users' browsing habits. The patterns that are obtained can be used in a range of application domains, such as Web personalization, Web recommendation, website/page enhancements, Web caching, and so on if they are correctly examined. Of course, the usage pattern analysis method chosen will depend much on the exact application being used to exploit the patterns and its eventual goals. The most intriguing area where using Web usage patterns can help a Web application achieve its end goals is prediction through pattern discovery.

A Recommender System is a special kind of classification and prediction system on the Web that is very much in demand because it has a wide range of applications in various e-commerce sites, news sites, entertainment applications, and so on.

Figure 9.5 shows the working of a Recommender System:

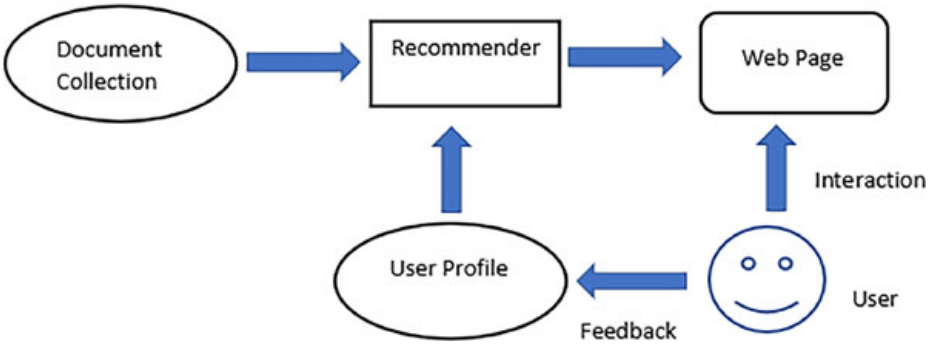


Figure 9.5: *Recommender System*

Web recommendation is an exciting technology that aims to anticipate user interests by delivering goods and services they require without their explicit request. Particularly in e-commerce, the development of recommender systems has taken on a significant amount of significance. By assisting customers in finding the necessary products, recommending related products, and fostering greater customer loyalty, such systems are intended to increase e-commerce sales in this context. Among these, Amazon stands out as one of the most well-known instances of an e-commerce site with a variety of integrated recommendation features that propose more products to a consumer based on his or her past purchases and browsing habits.

A technique called WebCF-DT uses data from Web usage, product purchases, and customer-related data to make product recommendations. Other significant e-commerce systems featuring recommendation features, such as personalized shopping and product suggestions that meet client needs, include E-bay, MovieFinder, and MovieLens.

In the literature, we find two basic approaches for recommendations:

- **Content-based recommendations:** In Content-based recommendations, items that are like past purchase history or preferences of the user are recommended. It uses supervised machine learning to train a classifier to distinguish between items that are interesting to the user and those that are not.
- **Collaborative filtering (or collaborative recommendations):** In Collaborative recommendation, the recommendations are made to the user based on the past liking of people of similar tastes and preferences. The basic assumption behind the algorithm is that users with similar interests have common preferences.

Building a content-based recommendation system

While designing a content-based recommendation system, at the first step, a profile for each item is required to be created. The profile represents the properties of

those items. A particular user is inferred from the user profile. Now, we are going to use these user profiles to recommend the items to the users from the user dataset.

Item profile

In a content-based recommendation system, we need to build a profile for each item, which contains the important properties of each item. For Example, If the song is an item, then its singer, musician, movie name/Album name, and genre are its important properties.

For creating Item Profile, first, we need to perform a TF-IDF vectorizer. A word's **term frequency** (**TF**) refers to how frequently it appears in a document, and a word's **inverse document frequency** (**IDF**) indicates how relevant that term is across the entire corpus.

User profile

A vector that encapsulates a user's preferences is the user profile. We use a utility matrix to express the relationship between the user and the object when building the user profile. The most accurate estimation we can make based on this data is to combine the profiles of the various things to determine which one the user prefers.

Conclusion

Web log data is a vast collection of data. The Web log data contains a lot of interesting trends. But without a pre-processing stage, it is exceptionally challenging to yield interesting patterns. The records are cleaned up during the pre-processing step, and noteworthy user patterns and session formation are discovered. In this chapter, we learnt that an essential task of Web usage mining applications is data pre-processing. Therefore, before using data mining techniques to extract user access patterns from Web logs, data must be analyzed. The data preparation process includes steps like data cleaning, user identification, session identification, path completion, and transaction identification. After that, the pre-processed data is prepared for additional pattern research and analysis.

Further, we have discussed that once the Pre-Processing stage is over, a suitable statistical or machine-learning model is applied to the processed data. The most used models for knowledge discovery, such as Association rule mining, Sequential learning, Clustering, Classification, and their possible use on Web usage data, have been discussed in detail. The predictions could be performed on the transaction pattern. Collaborative, as well as content-based filtering, can be performed for building intelligent Recommender Systems.

Points to remember

- Web usage mining is to discover interesting usage patterns from web data

using data mining techniques.

- The process of Web usage mining may be categorized into three stages Data collection and pre-processing, Pattern discovery, Pattern analysis.
- Data pre-processing consists of data cleaning, user identification, session identification, path completion and transaction identification.
- Pre-processed data and patterns lead to the formulation of models which are basically used as a primary input for many applications like web analytics, recommendations systems etc.
- Web data can be classified in to four classes – Usage data, Content data, Structure data, and User data.
- In Web Usage Mining, clustering is a technique for gathering comparable experiences. Clustering could be performed either by user clustering or by page clustering.
- Classification Mining is focused on creating a profile of things that fit into a certain group and are categorized in accordance with their similar characteristics.
- In a content-based recommendation system, we need to build a profile for each item and a vector to encapsulates user's preferences.

Multiple choice questions

1. Patterns that can be discovered from a given database are which type

    a. More than one type

    b. Multiple type always

    c. One type only

    d. No specific type

2. Which of the following is true for classification

    a. A subdivision of a set

    b. A measure of the accuracy

    c. The task of assigning a classification

    d. All of these

3. Which of the following forms of data mining assigns records to one of a predefined set of classes?

    a. Classification

    b. Clustering

    c. Both A and B

    d. None

4. Why are recommendation engines becoming popular?

   a. Users have lesser time, more options and face an information overload

   b. It is mandatory to have recommendation engine as per telecom rules

   c. It is better to recommend than ask user to search on mobile phones

   d. Users do not know what they want

5. What kind of information does a Recommendation Engine need for effective recommendations?

   a. Users' explicit interactions, such as information about their past activity, ratings, and reviews3

   b. Users' implicit interactions, such as the device they use for access, clicks on a link, location, and dates

   c. Other information about the profile, such as gender, age, or income levels

   d. All of the above

6. Which of the following is a collaborative filtering-based Recommendation Engine? Maximum value for Indegree.

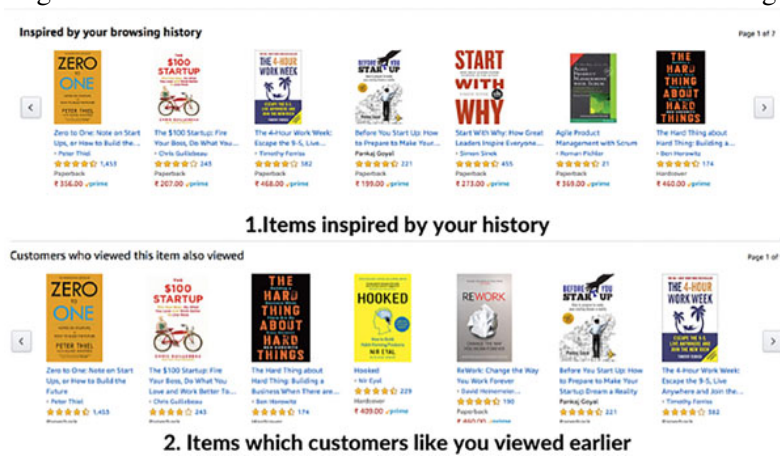


Figure 9.6: *Collaborative filtering-based recommendation system*

   a. Items inspired by your history

   b. Items that customers like you viewed earlier

   c. All of the above

   d. None of the above

7. In the example predicting the number of new-borns, the final number of total new-borns can be considered as the

   a. Features

   b. Observations

c. attributes
d. Outcome

8. K-means is an example of
    a. Association rule
    b. Clustering
    c. Regression
    d. Classification
9. Web Server Data includes
    a. IP Address
    b. Page reference
    c. Access Time
    d. All of the above
10. Clustering is known as
    a. Supervised Learning
    b. Unsupervised Learning
    c. Predictive Learning
    d. None of the above

Answers

Questions

1. Explain pre-processing on Web log data.
2. How knowledge discovery is performed.
3. What are the Recommender Systems. What are the requirements for designing a Recommender System?

Key terms

- Web usage mining
- Web usage data pre-processing
- Knowledge discovery
- Association rule
- Recommender system

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W