Pune Vidyarthi Griha's College of Engineering and Technology
&
G.K. Pate (Wani) Institute of Management, Pune
Approved by AICTE, DTE (CODE: 6274) Affiliated to SPPU, Pune | NAACThird Cycle 'A'

Artificial Intelligence and Data Science Department

Information Retrieval Lab Manual

Pune Vidyarthi Griha's College of Engineering and Technology
&
G.K. Pate (Wani) Institute of Management, Pune
Approved by AICTE, DTE (CODE: 6274) Affiliated to SPPU, Pune | NAACThird Cycle 'A'

FOR THE PROGRAMME

B. E. (ARTIFICIAL INTELLIGENCE AND DATA SCIENCE)-2020 Course

Institute's Vision Statement

"To achieve excellence in engineering education"

Institute's Mission Statement

- To satisfy all stakeholders
- To develop ethical, highly motivated engineering professionals with good human values, requisite skills and competencies.
- To adopt innovative teaching mechanisms.To
promote research culture
- To contribute to country's economic developments.
- To be responsive to changes In technology, socio-economic and environmental conditions

Artificial Intelligence and Data Science Department

Vision

"To become a center of excellence for education in Artificial Intelligence and Data Science with holistic development approach."

Mission

- To create a research based educational ecosystem by adopting interdisciplinary and multi-disciplinary approach
- To impart fundamental and advanced engineering knowledge to provide sustainable solutions to the industry and other stakeholders
- To nurture social and ethical values in students for the progress of society and nation

Program Educational Objectives (PEOs)

Graduate of the program will

PEO1 Apply problem solving skills to address the societal and environmental issues

PEO2 Practice Ethical AI and become Responsible Data Scientist

PEO3 Embrace lifelong learning to accommodate fast paced technical world of Artificial Intelligence

PEO4 Exhibit interpersonal and intra-personal skills to become a leader in the domain of AI&DS

PROGRAM OUTCOMES (POs)

Engineering graduates will possess:

PO1: Engineering Knowledge

Apply the knowledge of mathematics, science, Engineering fundamentals, and an Engineering specialization to the solution of complex Engineering problems.

PO2: Problem Analysis

Identify, formulate, review research literature, and analyze complex Engineering problems reaching substantiated conclusions using first principles of mathematics natural sciences, and Engineering sciences.

PO3: Design/Development of Solutions

Design solutions for complex Engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and Environmental considerations.

PO4: Conduct Investigations of Complex problems

Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern Tool Usage

Create, select, and apply appropriate techniques, resources, and modern Engineering and IT tools including prediction and modeling to complex Engineering activities with an understanding of the limitations.

PO6: The Engineer and Society

Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and Sustainability

Understand the impact of the professional Engineering solutions in societal and Environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics

Apply ethical principles and commit to professional ethics and responsibilities and norms of the Engineering practices.

PO9: Individual and Team Work

Function effectively as an individual, and as a member or leader in diverse teams, and multidisciplinary settings.

PO10: Communication

Communicate effectively on complex Engineering activities with the Engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions

PO11: Project Management and Finance

Demonstrate knowledge and understanding of the Engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary Environments.

PO12: Life-long Learning

Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAM SPECIFIC OUTCOMES (PSOs)

Program Specific Outcomes (PSO) AI&DS Engineering graduate will

- Demonstrate proficiency in collecting, cleaning, and analyzing diverse datasets using state-of the-art tools and techniques
- Exhibit the machine learning expertise and deep learning competency to design and train neural networks for various applications
- Adhere to professional standards and ethical principles in their AI and data science work,

List of Experiments

1. Write a program for pre-processing of a text document such as stop word removal, stemming.
2. Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using the standard Heart Disease Data Set (You can use Java/Python ML library classes/API. Data Cleaning and Preparation
3. Implement Agglomerative hierarchical clustering algorithm using appropriate dataset.
4. Implement Page Rank Algorithm. (Use python or beautiful soup for implementation).
5. Build the web crawler to pull product information and links from an e-commerce website.

Curriculum-Practical

Savitribai Phule Pune University
Fourth Year of Artificial Intelligence and Data Science (2020 Course)
417526:Computer Laboratory II: Information Retrieval

Teaching Scheme: **PR: 04 Hours/Week**	**Credit 02**	**Examination Scheme and Marks** **Term Work (TW): 50 Marks** **Practical (PR): 25 Marks**

Companion Course: Elective IV: Information Retrieval (417524(B))

Course Objectives:
- Understand the concepts of information retrieval and web mining
- Understand information retrieval process using standards available tools

Course Outcomes:
CO1: Apply various tools and techniques for information retrieval and web mining
CO2: Evaluate and analyze retrieved information

List of Assignments

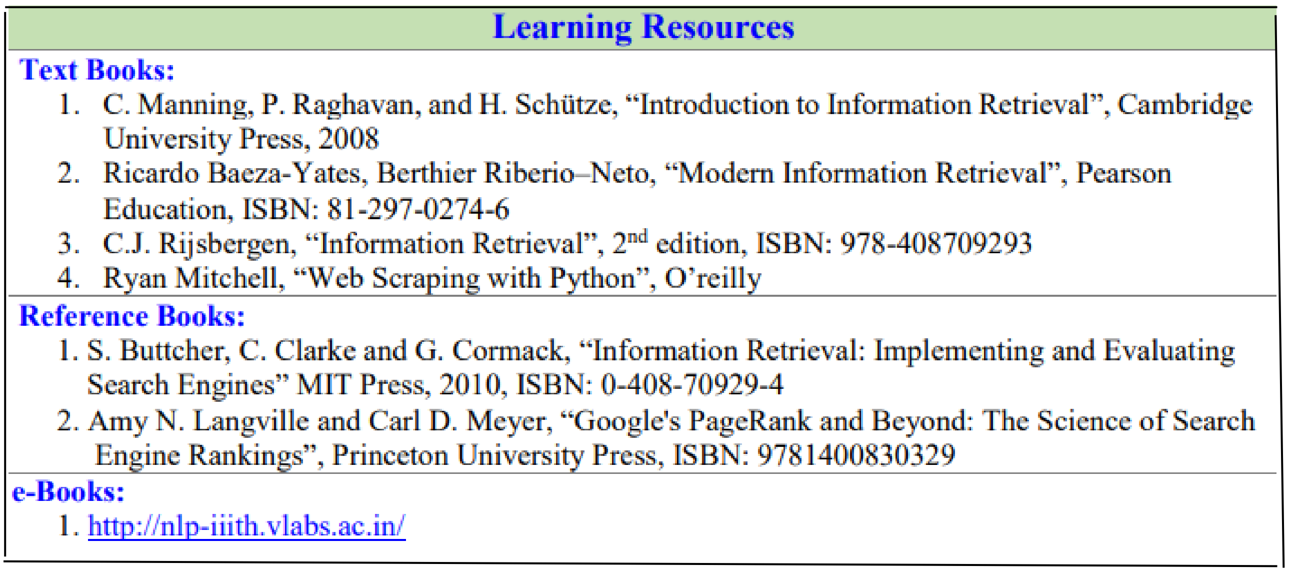
1. Write a program for pre-processing of a text document such as stop word removal, stemming.

2. Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using the standard Heart Disease Data Set (You can use Java/Python ML library classes/API.

3. Implement Agglomerative hierarchical clustering algorithm using appropriate dataset.

4. Implement Page Rank Algorithm. (Use python or beautiful soup for implementation).

5. Build the web crawler to pull product information and links from an e-commerce website.



Learning Resources

Text Books:

1. C. Manning, P. Raghavan, and H. Schütze, "Introduction to Information Retrieval", Cambridge University Press, 2008
2. Ricardo Baeza-Yates, Berthier Riberio–Neto, "Modern Information Retrieval", Pearson Education, ISBN: 81-297-0274-6
3. C.J. Rijsbergen, "Information Retrieval", 2nd edition, ISBN: 978-408709293
4. Ryan Mitchell, "Web Scraping with Python", O'reilly

Reference Books:

1. S. Buttcher, C. Clarke and G. Cormack, "Information Retrieval: Implementing and Evaluating Search Engines" MIT Press, 2010, ISBN: 0-408-70929-4
2. Amy N. Langville and Carl D. Meyer, "Google's PageRank and Beyond: The Science of Search Engine Rankings", Princeton University Press, ISBN: 9781400830329

e-Books:

1. http://nlp-iiith.vlabs.ac.in/

CO-PO Matrix

CO-PO Matrix:

The CO-PO Mapping Matrix

CO/ PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	1	1	2	3	2	-	-	-	-	-	-	1
CO2	1	1	2	3	2	-	-	-	-	-	-	1

Assessment Criterion

Sr. No.	Criteria	Marks
1	Attendance	5
2	Coding Efficiency	5
3	Timely Submission	5
4	Oral	5
5	Mock Practical	5
Total		25

Assignment 1

Title: Write a program for pre-processing of a text document such as stop word removal, stemming.

Problem Statement: Preprocessing a Text Document.

Dataset: Any Text document(s) provided for preprocessing.

Description: The goal is to create an algorithm that will prepare a text document for analysis. The pre-processing stages are as follows: stop word removal and stemming, which are both typical to natural language processing (NLP). Pre-processing is highly important when it comes to the removal of noise in data and enhancing the deployed machine learning procedure, in text data.

Tasks to Perform:

1. Load the text data by reading the content of the text file using Python’s file handling functions, storing it in a suitable data structure like a string or list of sentences.

2. Explore the structure of the text to identify special characters, numbers, or inconsistencies. Count words or sentences to get an overview of the content.

3. Remove stop words by utilizing libraries like NLTK or SpaCy. Stop words such as “the”, “is”, or “and” can be safely removed for most applications.

4. Apply stemming to reduce words to their root forms using the NLTK library’s Porter Stemmer or Snowball Stemmer. For example, “running” and “runs” will both be stemmed to “run.”

5. Store the cleaned text in a new data structure, such as a list of tokens or sentences, to facilitate further analysis.

6. Analyze the cleaned text by calculating metrics such as the most common words, sentence lengths, or unique terms.

7. Create visualizations like word clouds or bar charts to visualize word frequencies or other text features for gaining insights.

Prerequisite:

1. Basic knowledge of Python programming, including file handling, loops, and string manipulation.

2. Understanding of key text preprocessing concepts such as stop word removal and stemming in Natural Language Processing (NLP).

Theory:

Text Preprocessing:
Text preprocessing is a crucial step in Natural Language Processing (NLP) as it transforms raw text data into a structured format that can be used by machine learning algorithms. Raw text often contains unnecessary noise such as punctuation, numbers, and frequently occurring words that don't add much value for the task at hand. Without preprocessing, the performance of models can degrade due to the presence of such irrelevant or redundant information.

Stop Word Removal:
Stop words are common words that appear frequently in text but carry little meaning on their own. Words like "the," "and," or "is" are often considered stop words. By removing these words, you can reduce the dimensionality of the data and focus on the more meaningful words. Libraries like **NLTK** and **SpaCy** come with pre-defined lists of stop words that can be used for this purpose.

Stemming:
Stemming is a process of reducing words to their root form. For example, words like "running," "runner," and "ran" can all be reduced to the base form "run." This normalization helps group similar terms together, making the analysis more consistent. **Porter Stemmer**, **Snowball Stemmer**, and **Lancaster Stemmer** are widely used algorithms for stemming.

While stemming can improve the quality of the dataset, it can also sometimes result in over-stemming, where the root word becomes too generic (e.g., "connection" and "connect" both being reduced to "connect"). This is a trade-off that needs to be carefully managed depending on the task at hand.

Importance of Preprocessing:
Text data is inherently messy, and preprocessing ensures that the text becomes manageable for machine learning algorithms. Beyond stop word removal and stemming, preprocessing can include:

- Tokenization (splitting the text into individual words or phrases)
- Lowercasing all words to ensure consistency
- Handling punctuation and special characters
- Removing numerical data that may not be relevant

These steps help clean the text data, making it easier for algorithms to extract meaningful patterns and insights. Without preprocessing, text data might contain unnecessary noise, leading to inaccurate models or skewed results in analytics.

.

Conclusion:

In this lab, we explored essential techniques for text preprocessing, including stop word removal and stemming. These methods help clean and simplify textual data, ensuring that irrelevant information is minimized and meaningful patterns are preserved. Stop word removal reduces the noise in text by filtering out common words that don't contribute much to the analysis, while stemming helps standardize words by reducing them to their root form.

Assignment 2

Title: Diagnosis of Heart Patients using a Bayesian Network.

Problem Statement: Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using the standard Heart Disease Data Set (You can use Java/Python ML library classes/API.

Dataset: Heart Disease Data Set.

Description: The objective of this lab is to construct a Bayesian network using medical data and apply it to diagnose heart disease in patients. This will be demonstrated using the standard Heart Disease Dataset, leveraging Python's machine learning libraries to build and evaluate the model.

Tasks to Perform:

1. Load the Heart Disease Data Set. First, obtain the standard Heart Disease Data Set (available in CSV or other file formats). Load the data into Python using libraries like Pandas for easy manipulation..

2. Explore the data. Examine the structure of the dataset to identify key features like age, cholesterol level, blood pressure, etc. Check for missing values, inconsistencies, or outliers in the dataset that may require data cleaning.

3. Perform data preprocessing. Ensure the dataset is ready for input into the Bayesian network. This includes handling missing values, encoding categorical variables, or normalizing numerical features.

4.Construct a Bayesian network. Use a Python library like pgmpy or the BayesianNetworks module in scikit-learn. Create the network based on medical data, defining nodes and edges representing features and their relationships.

5.Train the Bayesian network. Split the dataset into training and testing sets to validate the model's performance. Estimate conditional probability distributions for each node based on the dataset.

6.Diagnose heart disease. Use the trained Bayesian network to make predictions. Based on input features such as age, cholesterol, and blood pressure, predict the likelihood of heart disease in a patient. Evaluate the model's accuracy using precision, recall, and F1-score.

7.Create visualizations. Represent the Bayesian network and its relationships visually using tools like NetworkX or Matplotlib. Generate confusion matrices or ROC curves to evaluate and visualize model performance.

Prerequisites:

1. **Basic Python Programming:** Familiarity with Python programming, data handling, and basic machine learning concepts is essential.

2. **Bayesian Networks:** A basic understanding of Bayesian networks, probabilistic graphical models, and their role in decision-making systems is necessary.

3. **Machine Learning Libraries:** Experience with Python ML libraries such as pgmpy, scikit-learn, Pandas, and Matplotlib will facilitate the implementation of the project.

Theory:

Bayesian networks, also known as belief networks, are powerful tools used in statistics and machine learning for modeling uncertainty and making probabilistic inferences. They are particularly useful in domains such as medicine, where many variables interact in complex ways, and understanding these relationships can significantly impact decision-making processes.

A Bayesian network consists of nodes and directed edges. Each node represents a random variable, which can be either discrete or continuous. The edges represent conditional dependencies between these variables. The structure of the network is typically defined based on domain knowledge or learned from data, allowing the model to capture causal relationships effectively.

In the context of medical diagnosis, a Bayesian network can represent various risk factors associated with heart disease, such as age, cholesterol levels, blood pressure, and lifestyle choices (e.g., smoking, exercise). By utilizing the Heart Disease Data Set, which contains historical data on patients and their heart health, we can train the network to estimate the conditional probabilities of having heart disease given specific values of these risk factors.

The training process involves calculating the conditional probability distributions for each node, based on observed data. For example, the probability of having high cholesterol given a certain age and smoking status can be computed, which allows the model to make informed predictions about new patients based on their medical attributes.

One of the key strengths of Bayesian networks is their ability to perform inference. After the model has been trained, it can be used to make predictions about the likelihood of a patient having heart disease. Given specific inputs (e.g., a 55-year-old male with high cholesterol), the network computes the probability of heart disease, enabling clinicians to make better-informed decisions about patient care.

Moreover, Bayesian networks can handle uncertainty in the data effectively. They allow for the incorporation of prior knowledge and can update probabilities as new information becomes available, making them adaptable in real-world scenarios where patient data may be incomplete or evolving.

The performance of the Bayesian network can be evaluated using various metrics, such as accuracy, precision, recall, and F1-score. These metrics help assess how well the model predicts heart disease and can guide adjustments to improve its performance.

In addition to making predictions, visualizing the Bayesian network can provide insights into the relationships between different risk factors. Using tools like NetworkX or Matplotlib, we can create graphical representations that show how different variables interact, helping to communicate findings to stakeholders.

Conclusion:

In this lab, we successfully constructed a Bayesian network to diagnose heart disease based on medical data from the Heart Disease Data Set. By utilizing probabilistic reasoning and the relationships among various risk factors, we were able to model the dependencies between different variables such as age, cholesterol levels, blood pressure, and lifestyle habits.

Assignment 3

Title: Implementing Agglomerative Hierarchical Clustering Algorithm

Problem Statement: Implement Agglomerative hierarchical clustering algorithm using appropriate dataset.

Dataset: CC.GENERAL.csv

Description: The cc_general.csv dataset contains credit card transactions data, typically used for financial analysis and modeling. It includes information about customer transactions and their corresponding credit card usage behavior. This dataset is widely used in credit scoring, customer segmentation, and behavioral analysis to understand spending habits and assess credit risk among customers.

Tasks to Perform:

1. Select an appropriate dataset for clustering analysis. Popular options include the Iris dataset, the Wine dataset, or any other dataset with multiple features conducive to clustering. Load the dataset into Python using libraries like Pandas.

2. Conduct exploratory data analysis (EDA) to understand the dataset's structure and characteristics. Examine key statistics, check for missing values, and visualize the distribution of features to inform your clustering approach.

3. Prepare the dataset for clustering by performing necessary preprocessing steps. This may include scaling numerical features to ensure they contribute equally to distance calculations. Techniques such as Standardization (Z-score normalization) or Min-Max scaling can be employed.

4. Use a Python library such as scikit-learn to implement the Agglomerative Hierarchical Clustering algorithm. Choose an appropriate distance metric (such as Euclidean distance) and linkage criteria (such as single, complete, or average linkage) to define how the clusters are formed.

5. Determine the optimal number of clusters using techniques such as the dendrogram plot and silhouette analysis. The dendrogram provides a visual representation of the hierarchical relationships between clusters, while silhouette scores quantify how similar an object is to its own cluster compared to other clusters.

6. Fit the Agglomerative Hierarchical Clustering model to the preprocessed data and predict the clusters for each data point. Assign cluster labels to the original dataset based on the clustering results.

7. Create visualizations to represent the clustered data. Utilize 2D scatter plots to show how data points are grouped. If necessary, reduce the dimensionality of the data using techniques like PCA (Principal Component Analysis) for better visualization of clusters.

Prerequisites:

1. Basic Python Programming: Familiarity with Python programming, data manipulation, and data visualization is essential for this lab.
2. Clustering Concepts: A fundamental understanding of clustering algorithms, specifically hierarchical clustering, is necessary.
3. Machine Learning Libraries: Experience with Python libraries such as Pandas, NumPy, Matplotlib, and scikit-learn will facilitate the implementation of the assignment.

Theory:

Agglomerative Hierarchical Clustering is a bottom-up clustering approach where each data point starts as its own individual cluster. The algorithm iteratively merges clusters based on their similarity until a specified number of clusters is formed or all points are merged into a single cluster.

The process begins with each data point as a singleton cluster. The algorithm calculates the distance between every pair of clusters and merges the closest ones. This process continues, creating a hierarchy of clusters that can be visualized using a dendrogram. A dendrogram is a tree-like diagram that illustrates the arrangement of the clusters formed at each step, providing insight into the data's structure.

One of the strengths of agglomerative clustering is its ability to create a hierarchy, which allows for different levels of granularity in clustering. The choice of linkage criteria—single, complete, average, or ward—affects how distances are computed between clusters, which influences the final clustering result.

Single linkage considers the minimum distance between points in two clusters, while complete linkage uses the maximum distance. Average linkage calculates the average distance between all pairs of points in two clusters, and ward linkage minimizes the total within-cluster variance. These criteria should be chosen based on the characteristics of the dataset and the desired clustering behavior.

To evaluate the quality of the clustering results, metrics such as silhouette scores can be used. Silhouette scores measure how well each data point is clustered, indicating whether it is appropriately grouped with similar points or if it might be better suited to another cluster. Values closer to +1 indicate better-defined clusters, while values near 0 suggest overlapping clusters.

Visualization of the clustering results is essential to interpret and validate the model's effectiveness. By plotting the data points colored by their cluster assignments, patterns and relationships within the data become more apparent. Dimensionality reduction techniques such as PCA can assist in visualizing high-dimensional datasets in two or three dimensions.

Key Steps in Agglomerative Clustering:

1. Distance Calculation: Different distance metrics can be utilized to calculate the proximity between data points. Common metrics include:

- Euclidean Distance: Measures the straight-line distance between two points in Euclidean space.
- Manhattan Distance: Calculates the distance based on the sum of the absolute differences of their Cartesian coordinates.
- Cosine Similarity: Measures the cosine of the angle between two non-zero vectors in an inner product space.

2. Linkage Criteria: The choice of linkage criteria affects how clusters are formed and can significantly impact the resulting clusters. Some common linkage methods are:

- **Single Linkage:** Merges clusters based on the shortest distance between points in the clusters.
- **Complete Linkage:** Merges clusters based on the longest distance between points in the clusters.
- **Average Linkage:** Considers the average distance between all pairs of points in two clusters.
- **Ward's Linkage:** Merges clusters to minimize the total within-cluster variance.

Conclusion :

By the end of this lab, you should have successfully implemented the Agglomerative Hierarchical Clustering algorithm, assigned cluster labels to data points, and visualized the clustering results. The analysis should provide insights into the underlying patterns in the dataset and demonstrate your understanding of hierarchical clustering concepts.

Assignment 4

Title: Implementing the PageRank Algorithm

Problem Statement: Implement Page Rank Algorithm. (Use python or beautiful soup for implementation).

Description: The objective of this lab is to implement the PageRank algorithm using Python. This exercise will help you understand how web pages are ranked based on their link structure, simulating a simplified version of how search engines determine the importance of web pages.

Tasks to Perform:

1. Select or create a dataset that represents a web graph. The dataset should consist of web pages as nodes and hyperlinks as edges connecting these nodes. You can use a simple graph with a few pages and their links or a larger dataset if available.

2. Use Python to represent the web graph. This can be done using a dictionary or an adjacency list to illustrate the connections between web pages.

3. Implement the PageRank algorithm. The PageRank algorithm works by iteratively calculating the rank of each page based on the ranks of the pages that link to it. The basic formula for PageRank is as follows:

$$PR(A) = \frac{1-d}{N} + d\sum_{i=1}^{k} \frac{PR(B_i)}{C(B_i)}$$

where:

- PR(A)PR(A)PR(A) is the PageRank of page A.
- ddd is the damping factor (usually set around 0.85).
- NNN is the total number of pages.
- kkk is the number of pages linking to page A.
- PR(Bi)PR(B_i)PR(Bi) is the PageRank of page B linking to page A.
- C(Bi)C(B_i)C(Bi) is the number of outbound links from page B.

4. Iterate the PageRank calculation until the ranks converge. You may set a tolerance level to determine when to stop iterating, ensuring that the PageRank values change minimally between iterations.
5. Analyze the results by sorting the pages based on their PageRank scores. Identify the top-ranked pages and discuss their significance in the context of the dataset.
6. Visualize the web graph and the PageRank scores. Use libraries like Matplotlib or NetworkX to create visual representations of the web graph and highlight the ranks of different pages.

Prerequisites:

1. Basic Python Programming: Familiarity with Python programming, data structures, and libraries is essential for this lab.
2. Graph Theory Concepts: A fundamental understanding of graph theory, including nodes and edges, will help in representing and analyzing the web graph.
3. Python Libraries: Experience with libraries such as Beautiful Soup (for web scraping, if needed), NumPy, and Matplotlib will facilitate the implementation of the assignment.

Theory:

The PageRank algorithm, developed by Larry Page and Sergey Brin, is a pivotal algorithm used by search engines to rank web pages in their search results. It operates on the principle that the importance of a web page can be determined by the quantity and quality of links pointing to it. The more high-quality links a page receives, the higher its PageRank score.

In its simplest form, the PageRank algorithm treats the web as a directed graph, where each web page is represented as a node and hyperlinks between pages as directed edges. The algorithm uses an iterative approach to calculate the PageRank of each page, which reflects its relative importance in the web graph.

Key Components of PageRank:

1. **Damping Factor:** The damping factor ddd accounts for the probability that a user will continue clicking links or will jump to a random page. It is typically set to 0.85, implying that 15% of the time, a user randomly selects a page.
2. **Convergence Criteria:** To ensure the algorithm terminates, a convergence criterion is established. This criterion may involve checking if the change in PageRank values between iterations is below a predefined threshold.
3. **Matrix Representation:** The web graph can be represented as a stochastic matrix, where each entry indicates the probability of transitioning from one page to another. The PageRank vector can be calculated using matrix operations, leveraging the properties of linear algebra.

Conclusion:

This lab demonstrated the implementation of the PageRank algorithm to rank web pages based on their link structure. Key concepts of graph theory, iterative calculations, and the role of damping factors in ranking were explored. Understanding how the PageRank algorithm functions is crucial in information retrieval and web search, providing foundational knowledge applicable to more complex scenarios in data analysis and search engine optimization.

Assignment 5

Title: Building a Web Crawler for E-commerce Product Information

Problem Statement: Build the web crawler to pull product information and links from an e-commerce website.

Description: The objective of this lab is to develop a web crawler that extracts product information and links from an e-commerce website. This exercise will provide practical experience in web scraping, data extraction, and handling structured data.

Tasks to Perform:

1. Select an e-commerce website to crawl. Ensure that the website allows web scraping by checking its `robots.txt` file and terms of service.

2. Use Python and libraries such as Beautiful Soup and Requests to build the web crawler. The Requests library will handle HTTP requests, while Beautiful Soup will parse the HTML content.

3. Implement the crawler to navigate the website and extract relevant product information. This includes identifying HTML elements that contain product names, prices, descriptions, and links. Typically, this information can be found within specific tags or classes in the HTML structure.

4. Store the extracted data in a structured format. Options for storage include CSV files, JSON files, or databases like SQLite or MongoDB, depending on the requirements.

5. Handle pagination if the website lists multiple pages of products. The crawler should be able to navigate through these pages to gather data from all available products.

6. Ensure the crawler respects the website's rate limits and does not overload the server with requests. Implement delays between requests if necessary.

7. Test the crawler and verify that the data extracted is accurate and complete. Output sample results to confirm that the desired product information has been successfully retrieved.

Prerequisites:

1. Basic Python Programming: Familiarity with Python programming and data structures is essential for this lab.

2. Web Scraping Concepts: Understanding the principles of web scraping, including HTML structure, HTTP requests, and data parsing, will aid in the effective implementation of the crawler.

3. Python Libraries: Experience with libraries such as Beautiful Soup and Requests is necessary for constructing the web crawler and handling the extracted data.

Theory:

Web scraping is a technique used to extract information from websites. It involves sending requests to a web server, retrieving HTML content, and parsing it to extract relevant data. In the context of e-commerce, web scraping can be invaluable for gathering product information, price comparisons, market research, and monitoring competitor activities.

Key Components of Web Crawling:

1. **HTTP Requests:** The crawler uses HTTP methods (GET, POST) to communicate with the web server. The Requests library simplifies this process by handling various aspects of HTTP requests and responses.

2. **HTML Parsing:** After retrieving the HTML content of a webpage, parsing is necessary to extract specific data. Beautiful Soup provides tools to navigate the parse tree and find desired elements based on their tags, classes, or attributes.

3. **Data Storage:** Once the data is extracted, it must be stored in a structured format for further analysis. Common formats include CSV, JSON, and databases. Choosing the appropriate format depends on the intended use of the data.

4. **Respecting Website Policies:** It is crucial to respect the website's `robots.txt` file and comply with its terms of service. Overloading the server with requests can lead to IP bans or legal issues. Implementing delays and rate limiting is essential to avoid these problems.

Conclusion:

This lab demonstrated the construction of a web crawler to extract product information from an e-commerce website. Key concepts of web scraping, including handling HTTP requests, parsing HTML, and storing data, were explored. Understanding how to effectively gather data from online sources is essential for various applications in data analysis, market research, and business intelligence.