PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE
ACADEMIC YEAR: 2024-25

DEPARTMENT OF COMPUTER ENGINEERING DEPARTMENT

CLASS: B.E. **SEMESTER: I**

SUBJECT: LP-IV

ASSIGNMENT NO.	A1
TITLE	**Compute similarity between two text documents.**
PROBLEM STATEMENT /DEFINITION	Write a program to compute similarity between two text documents.
OBJECTIVE	To compute similarity between two text documents by making use of the **Vector Space** model. To understand the working of the **Vector Space** model.
OUTCOME	Students will be able to - • Compute the similarity between two text documents. • Understand the working of the **Vector Space** model.
S/W PACKAGES AND HARDWARE APPARATUS USED	Windows 10 (64-bit), Intel I5 4GB RAM 256 GB SSD, Python 3.9.0, VS Code
REFERENCES	1. C.J. Rijsbergen, &quot;Information Retrieval&quot;, (http://www.dcs.gla.ac.uk/Keith/Preface.html) 2. W.R. Hersh, ―Information Retrieval: A Health and Biomedical Perspective‖, Springer, 2002. 3. G. Kowalski, M.T. Maybury. &quot;Information storage and Retrieval System&quot; , Springer, 2005
STEPS	Refer to theory, algorithm, test input, test output
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Test cases 10. Conclusion/Analysis

Prerequisites:

Concepts related Theory:

Vector Space model

Vector space model or term vector model is an algebraic model for representing text documents (and any objects, in general) as vectors of identifiers (such as index terms). It is used in information filtering, information retrieval, indexing and relevancy rankings. Its first use was in the SMART Information Retrieval System.

Documents and queries are represented by vectors as follows:

$\mathbf{d_j} = (\mathbf{w_{1,j}} , \mathbf{w_{2,j}}, \ldots\ldots\ldots \mathbf{w_{n,j}})$

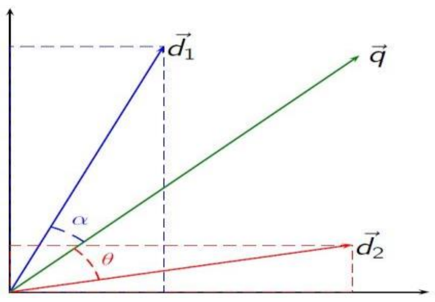
$\mathbf{q} = (\mathbf{w_{1,j}} , \mathbf{w_{2,j}}, \ldots\ldots\ldots \mathbf{w_{n,j}})$

Our aim is to find the weight vector based on the frequency or occurrences of terms.

The definition of term depends on the application. Typically terms are single words, keywords, or longer phrases. If words are chosen to be the terms, the dimensionality of the vector is the number of words in the vocabulary (the number of distinct words occurring in the corpus).

The correlation between the document $\mathbf{d_j}$ and the query $\mathbf{q}$ is given as the cosine of the angle between two vectors.

$$\cos(d_j, q) = \frac{\mathbf{d_j} \cdot \mathbf{q}}{\|\mathbf{d_j}\| \, \|\mathbf{q}\|} = \frac{\sum_{i=1}^{N} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{N} w_{i,j}^2}\sqrt{\sum_{i=1}^{N} w_{i,q}^2}}$$



If the value of cos(d_j, q) is less than the document is more similar and vice versa then we can get a degree of similarity between 0 and 1. A cosine value of zero means that the query and document vector are orthogonal and have no match (i.e. the query term does not exist in the document being considered) while a cosine value of 1 means that the query and document vector are exactly identical.

Algorithm:

1. Read the text file.
2. Tokenizing the lines into words.
3. Removing the punctuation marks and stop words.
4. Counting the frequency of each word in the query and document.
5. Taking the dot product of the query vector and the document vector.
6. Finding the angle value by taking the cosine inverse of the dot product.

Conclusion: We have successfully learned and implemented a vector space model and found similarity between the two text documents.

Review Questions:

Q1. What is text Summarization?

Q2. What is tokenization?

Q3. What do you mean by Text Extraction and Cleanup?

Q4. What are the steps to follow when building a text classification system?

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2023-24

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: B.E. **SEMESTER: I**

SUBJECT: LP-IV

ASSIGNMENT NO.	A2
TITLE	**Page Rank Algorithm**
PROBLEM STATEMENT /DEFINITION	Implement **Page Rank Algorithm**.
OBJECTIVE	To perform ranked retrieval of documents by assigning page ranks to each document as per the importance of the document.
OUTCOME	Students will be able to understand: • Perform ranked retrieval of the documents according to their importance by assigning them page ranks. • How Page Rank Algorithm works and used in various search engines.
S/W PACKAGES AND HARDWARE APPARATUS USED	Python 3.9.0, VS Code, Windows (64-bit), Intel I5 4GB RAM
REFERENCES	1. C.J. Rijsbergen, &quot;Information Retrieval&quot;, (http://www.dcs.gla.ac.uk/Keith/Preface.html) 2. W.R. Hersh, ―Information Retrieval: A Health and Biomedical Perspective‖, Springer, 2002. 3. G. Kowalski, M.T. Maybury. &quot;Information storage and Retrieval System&quot; , Springer, 2005
STEPS	Refer to theory, algorithm, test input, test output
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Test cases 10. Conclusion/Analysis

Prerequisites:

Concepts related Theory:

Page Rank Algorithm

PageRank (PR) is an algorithm used by Google Search to rank web pages in their search engine results. It is named after both the term "web page" and co-founder Larry Page. PageRank is a way of measuring the importance of website pages.

PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

PageRank is a link analysis algorithm and it assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set. The algorithm may be applied to any collection of entities with reciprocal quotations and references. The numerical weight that it assigns to any given element E is referred to as the PageRank of E and denoted by PR(E).

Assume a small universe of four web pages: **A**, **B**, **C**, and **D**. Links from a page to itself are ignored. Multiple outbound links from one page to another page are treated as a single link. The initial value of the web pages can be assumed to be anything between 0 and 1, let us say 0.25.

If the only links in the system were from pages **B**, **C**, and **D** to **A**, each link would transfer 0.25 PageRank to **A** upon the next iteration, for a total of 0.75.

$$PR(A) = PR(B) + PR(C) + PR(D).$$

Suppose instead that page **B** had a link to pages **C** and **A**, page **C** had a link to page **A**, and page **D** had links to all three pages. Thus, upon the first iteration, page **B** would transfer half of its existing value (0.125) to page **A** and the other half (0.125) to page **C**. Page **C** would transfer all of its existing value (0.25) to the only page it links to, **A**. Since **D** had three outbound links, it would transfer one third of its existing value, or approximately 0.083, to **A**. At the completion of this iteration, page **A** will have a PageRank of approximately 0.458.

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1} + \frac{PR(D)}{3}.$$

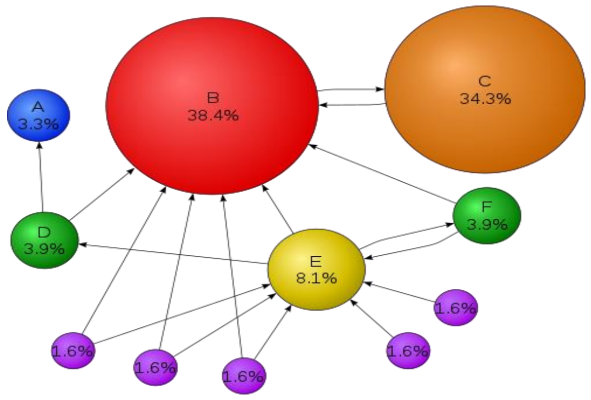
In other words, the PageRank conferred by an outbound link is equal to the document's own PageRank score divided by the number of outbound links **L( )**.

$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)}.$$

In the general case, the PageRank value for any page **u** can be expressed as:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

i.e. the PageRank value for a page **u** is dependent on the PageRank values for each page **v** contained in the set $\mathbf{B_u}$ (the set containing all pages linking to page **u**), divided by the number $L(v)$ of links from page **v**.



Above is the simple illustration of the Page Rank algorithm. The percentage shows the perceived importance, and the arrows represent hyperlinks.

Algorithm:

Iteration 0: Initialize all ranks to be 1/(number of total pages).

Iteration 1: For each page u, update u's rank to be the sum of each incoming page v's rank from the previous iteration, divided by the number total number of links from page v.

Conclusion: Page Rank Algorithm is implemented successfully using python programming language for ranked retrieval of documents.

Review Questions:

Q1. What are page speed and its role in SEO?

Q2. What are meta tags? Name the important ones and their character limits.

Q3. What is a landing page?

Q4. Explain – spiders, crawlers and robots.

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2023-24

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: B.E. **SEMESTER: I**

SUBJECT: LP-IV

ASSIGNMENT NO.	A3
TITLE	**Stop Word Removal**
PROBLEM STATEMENT /DEFINITION	Write a program for pre-processing of a Text Document: Stop Word Removal.
OBJECTIVE	• To learn and understand the process of text pre-processing. • To understand stop word removal process.
OUTCOME	• Implement stop word removal to remove frequently occurring non-important words like a, an, the, in, from, etc. • Implement pre-processing of text document for information retrieval.
S/W PACKAGES AND HARDWARE APPARATUS USED	64-bit Ubuntu OS, 8GB RAM(Recommended), Jupyter Notebook, Programming Language: Python
REFERENCES	1. C.J. Rijsbergen, &quot;Information Retrieval&quot;, (http://www.dcs.gla.ac.uk/Keith/Preface.html) 2. W.R. Hersh, ―Information Retrieval: A Health and Biomedical Perspective‖, Springer, 2002. 3. G. Kowalski, M.T. Maybury. &quot;Information storage and Retrieval System&quot; , Springer, 2005
STEPS	Refer to theory, algorithm, test input, test output
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Test cases 10. Conclusion/Analysis

Prerequisites:

Concepts related Theory:

Text Pre-Processing: Text data derived from natural language is unstructured and noisy.

Text pre-processing involves transforming text into a clean and consistent format that can then be fed into a model for further analysis and learning. Text pre-processing techniques may be general so that they are applicable to many types of applications, or they can be specialized for a specific task.

Steps in text pre-processing are:

1. **) Segmentation**
2. **) Tokenization**
3. **) Change Case**
4. **) Spell Correction**
5. **) Stop Words Removal**
6. **) Stemming**
7. **) Lemmatization**

Segmentation

Segmentation involves breaking up text into corresponding sentences. While this may seem like a trivial task, it has a few challenges. For example, in the English language, a period normally indicates the end of a sentence, but many abbreviations, including "Inc.," "Calif.," "Mr.," and "Ms.," and all fractional numbers contain periods and introduce uncertainty unless the end-of-sentence rules accommodate those exceptions.

Tokenization

The tokenization stage involves converting a sentence into a stream of words, also called "tokens." Tokens are the basic building blocks upon which analysis and other methods are built.

Many NLP toolkits allow users to input multiple criteria based on which word boundaries are determined. For example, you can use a whitespace or punctuation to determine if one word has ended and the next one has started. Again, in some instances, these rules might fail. For example, *don't*, *it's*, etc. are words themselves that contain punctuation marks and have to be dealt with separately.

Change Case

Changing the case involves converting all text to lowercase or uppercase so that all word strings follow a consistent format. Lowercasing is the more frequent choice in NLP software.

Spell Correction

Many NLP applications include a step to correct the spelling of all words in the text.

Stop-Words Removal

"Stop words" are frequently occurring words used to construct sentences. In the English language, stop words include *is, the, are, of, in,* and *and.* For some NLP applications, such as document categorization, sentiment analysis, and spam filtering, these words are redundant, and so are removed at the pre-processing stage.

Stemming

The term *word stem* is borrowed from linguistics and used to refer to the base or root form of a word. For example, *learn* is a base word for its variants such as *learn, learns, learning,* and *learned.*

Stemming is the process of converting all words to their base form, or stem. Normally, a lookup table is used to find the word and its corresponding stem. Many search engines apply stemming for retrieving documents that match user queries. Stemming is also used at the pre-processing stage for applications such as emotion identification and text classification.

Lemmatization

Lemmatization is a more advanced form of stemming and involves converting all words to their corresponding root form, called "lemma." While stemming reduces all words to their stem via a lookup table, it does not employ any knowledge of the parts of speech or the context of the word. This means stemming can't distinguish which meaning of the word *right* is intended in the sentences "Please turn right at the next light" and "She is always right."

The stemmer would stem *right* to *right* in both sentences; the lemmatizer would treat *right* differently based upon its usage in the two phrases.

A lemmatizer also converts different word forms or inflections to a standard form. For example, it would convert *less* to *little*, *wrote* to *write*, *slept* to *sleep*, etc.

A lemmatizer works with more rules of the language and contextual information than does a stemmer. It also relies on a dictionary to look up matching words. Because of that, it requires more processing power and time than a stemmer to generate output. For these reasons, some NLP applications only use a stemmer and not a lemmatizer.

NLTK: NLTK is one of the libraries provided by python for removal of stop words. You can find a list of stop words in the corpus module. To remove stop words, you can divide your text into words and then remove the word if it exists in the list of stop words provided by NLTK.

Algorithm:

The algorithm is implemented as below given steps.

Step 1: The target document text is tokenized and individual words are stored in array.

Step 2: A single stop word is read from stopword list.

Step 3: The stop word is compared to target text in form of array using sequential search technique.

Step 4: If it matches , the word in array is removed , and the comparison is continued till length of array.
Step 5: After removal of stopword completely, another stopword is read from stopword list and again algorithm follows step 2. The algorithm runs continuously until all the stopwords are compared.

Step 6: Resultant text devoid of stopwords is displayed, also required statistics like stopword removed, no. of stopwords removed from target text, total count of words in target text, count of words in resultant text, individual stop word count found in target text is displayed.

Conclusion: Pre-processing of text documents was done and the process of stop word removal was understood using NLTK library in python.

Review Questions:

Q1. What are stemming and lemmatization?

Q2. What is count vectorization?

Q3. What is word embedding?

Q4. What do you mean by Lemmatization in NLP?

Q5. What are stop words?

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2023-24

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: B.E. **SEMESTER: I**

SUBJECT: LP-IV

ASSIGNMENT NO.	A4
TITLE	**Map Reduce Program**
PROBLEM STATEMENT /DEFINITION	Write a **Map Reduce** program to count the number of occurrences of each alphabetic character in the given dataset. The count of each letter should be case-insensitive.
OBJECTIVE	Using Map Reduce processing technique to count the number of occurrences of each alphabetic character in the given document.
OUTCOME	Students will be able to: • Implement the Map Reduce program to count the occurrences of each alphabetic character in the given dataset. • Understand the working of the Map Reduce program.
S/W PACKAGES AND HARDWARE APPARATUS USED	Python 3.9.0, VS Code, Intel I5 8 GB RAM, Windows 10 (64-bit) OS
REFERENCES	1. C.J. Rijsbergen, &quot;Information Retrieval&quot;, (http://www.dcs.gla.ac.uk/Keith/Preface.html) 2. W.R. Hersh, ―Information Retrieval: A Health and Biomedical Perspective‖, Springer, 2002. 3. G. Kowalski, M.T. Maybury. &quot;Information storage and Retrieval System&quot; , Springer, 2005
STEPS	Refer to theory, algorithm, test input, test output
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Test cases 10. Conclusion/Analysis

Prerequisites:

Concepts related Theory:

Map Reduce

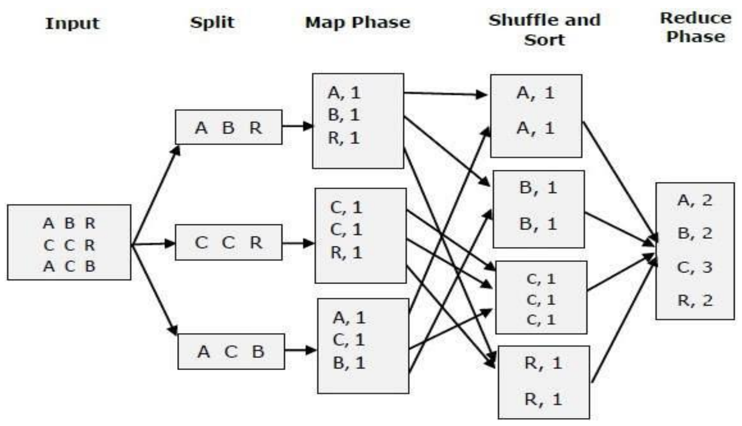
A MapReduce program is composed of a map procedure, which performs filtering and sorting (such as sorting students by first name into queues, one queue for each name), and a reduce method, which performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "MapReduce System" (also called "infrastructure" or "framework") orchestrates the processing by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance.

A MapReduce framework (or system) is usually composed of three operations (or steps):

1. **Map:** each worker node applies the map function to the local data, and writes the output to a temporary storage. A master node ensures that only one copy of the redundant input data is processed.

2. **Shuffle:** worker nodes redistribute data based on the output keys (produced by the map function), such that all data belonging to one key is located on the same worker node.

3. **Reduce:** worker nodes now process each group of output data, per key, in parallel.



Above diagram shows the working process of a MapReduce program.

Algorithm:

- The map takes data in the form of pairs and returns a list of <key, value> pairs. The keys will not be unique in this case.
- Using the output of Map, sort and shuffle are applied by the Hadoop architecture. This sort and shuffle acts on these list of <key, value> pairs and sends out unique keys and a list of values associated with this unique key <key, list(values)>.
- An output of sort and shuffle sent to the reducer phase. The reducer performs a defined function on a list of values for unique keys, and Final output <key, value> will be stored/displayed.

Conclusion: Successfully learned and implemented Map Reduce function to count the number of occurrences of each alphabetic character in the given dataset.

Review Questions:

Q1. What are the main components of MapReduce Job?

Q2. What is Shuffling and Sorting in MapReduce?

Q3. What is Text Input Format?

Q4. What is a MapReduce Combiner?

Q5. What are the parameters of mappers and reducers?

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2023-24

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: B.E. **SEMESTER: I**

SUBJECT: LP-IV

ASSIGNMENT NO.	A5
TITLE	**Web Crawling**
PROBLEM STATEMENT /DEFINITION	Write a simple program to implement a simple web crawler.
OBJECTIVE	• Learn and understand the concept of web crawling and web scraping. • Build a simple web crawler bot.
OUTCOME	Students will be able to: • Differentiate between web crawling and web scraping. • Implement a simple web crawler / spiderbot.
S/W PACKAGES AND HARDWARE APPARATUS USED	64- bit Ubuntu OS / Windows 10 OS, 8 GB RAM, Python 3.9.0, pip or condo to install scrapy library provided by Python
REFERENCES	1. C.J. Rijsbergen, &quot;Information Retrieval&quot;, (http://www.dcs.gla.ac.uk/Keith/Preface.html) 2. W.R. Hersh, ―Information Retrieval: A Health and Biomedical Perspective‖, Springer, 2002. 3. G. Kowalski, M.T. Maybury. &quot;Information storage and Retrieval System&quot; , Springer, 2005
STEPS	Refer to theory, algorithm, test input, test output
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Test cases 10. Conclusion/Analysis

Prerequisites:

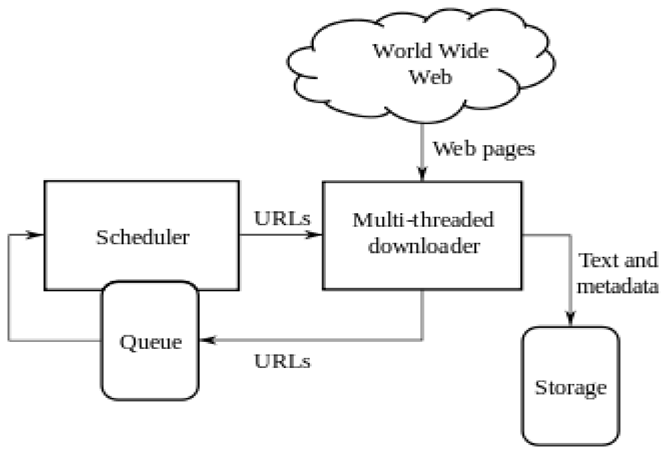
Concepts related Theory:

Web Crawler

A Web crawler, sometimes called a spider or spiderbot and often shortened to crawler, is an Internet bot that systematically browses the World Wide Web and that is typically operated by search engines for the purpose of Web indexing (web spidering).

Web search engines and some other websites use Web crawling or spidering software to update their web content or indices of other sites' web content. Web crawlers copy pages for processing by a search engine, which indexes the downloaded pages so that users can search more efficiently.

Crawlers consume resources on visited systems and often visit sites unprompted. Issues of schedule, load, and "politeness" come into play when large collections of pages are accessed. Mechanisms exist for public sites not wishing to be crawled to make this known to the crawling agent. For example, including a robots.txt file can request bots to index only parts of a website, or nothing at all.



Architecture of a Web crawler

How does a Web Crawler works?

A Web crawler starts with a list of URLs to visit. Those first URLs are called the seeds. As the crawler visits these URLs, by communicating with web servers that respond to those URLs, it identifies all the hyperlinks in the retrieved web pages and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are recursively visited according to a set of policies. If the crawler is performing archiving of websites (or web archiving), it copies and saves the information as it goes. The archives are usually stored in such a way they can be viewed, read and navigated as if they were on the live web, but are preserved as 'snapshots'.

The archive is known as the repository and is designed to store and manage the collection of web pages. The repository only stores HTML pages and these pages are stored as distinct files. A repository is similar to any other system that stores data, like a modern-day database. The only difference is that a repository does not need all the functionality offered by a database system. The repository stores the most recent version of the web page retrieved by the crawler.

Difference between Web Scraping and Web Crawling:

Web scraping is the process of using bots to extract content and data from a website.

Unlike screen scraping, which only copies pixels displayed on screen, web scraping extracts underlying HTML code and, with it, data stored in a database. The scraper can then replicate entire website content elsewhere.

Web scraping is used in a variety of digital businesses that rely on data harvesting. Legitimate use cases include:

- Search engine bots crawling a site, analyzing its content and then ranking it.
- Price comparison sites deploying bots to auto-fetch prices and product descriptions for allied seller websites.
- Market research companies using scrapers to pull data from forums and social media (e.g., for sentiment analysis.)

Algorithm:

- Step 1: Send an HTTP request to the URL of the webpage. It responds to your request by returning the content of web pages.
- Step 2: Parse the webpage. A parser will create a tree structure of the HTML as the webpages are intertwined and nested together. A tree structure will help the bot follow the paths that we created and navigate through to get the information.
- Step 3: Using the Python library to search the parse tree.

Conclusion:

1. This assignment helped us to understand the concept of web crawling.
2. Successfully implemented a simple web crawler with scrapy utility library provided by python.

Review Questions:

Q1**.** How can data scraped from the internet be used for business intelligence, analytics, or research purposes?

Q2. What are the pros and cons of web scraping?

Q3. Why Selenium is often preferred over other tools for web scraping?

Q4. What are the different types of web scraping techniques that can be used?

Q5. What are the main challenges faced while writing a web crawler?