***Group A***

***Assignment-1 : Conflation Algorithm***

**Input:**

package com.cl.conflation;

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

public class Conflation {

    public static ArrayList<String> stopList = new ArrayList<String>();

    public static ArrayList<String> removestopList = new ArrayList<String>();

    public static String suffixes[] = { "able", "ing", "ion", "y", "ment" };

    public static String stopwords[] = { "i", "big", "am", "m", "a",

                                         "we", "are", "it", "of", "this", "and", "is", "to",

                                         "at", "in", "was", "with", "doing", "It", "not", "our" };

    public static void main(String[] args) {

        InputStreamReader st = new InputStreamReader(System.in);

        BufferedReader buff = new BufferedReader(st);

        String fname = "";

        System.out.println("Enter a filename:");

        try {

            fname = buff.readLine();

        } catch (IOException e) {

            e.printStackTrace();

        }

        conflation(fname);

    }

    public static void conflation(String fname) {

        BufferedReader buff;

        int i = 0, j = 0;

        try {

            buff = new BufferedReader(new FileReader(fname));

            int flag = 0;

            String line = "";

            line = buff.readLine();

            String[] buffer = line.split(" ");

            for (i = 0; i < buffer.length; i++) {

                flag = 0;

                if (buffer[i].endsWith(".")) {

                    buffer[i] = buffer[i].replace(".", "");

                }

                for (j = 0; j < stopwords.length; j++) {

                    if (buffer[i].equals(stopwords[j])) {

                        stopList.add(buffer[i]);

                        flag = 1;

                        break;

                    }

                }

                if (flag != 1 && !buffer[i].equals(null)) {

                    removestopList.add(buffer[i]);

                }

            }

            System.out.println("\n--------------After Removing Stop Words-----------------");

            for (int k = 0; k < removestopList.size(); k++) {

                System.out.println(removestopList.get(k));

            }

            suffixesString(removestopList);

            countFrequency(removestopList);

        } catch (FileNotFoundException e) {

            e.printStackTrace();

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    private static void countFrequency(ArrayList<String> removestopList) {

        // Mapping of String->Integer (word -> frequency)

        System.out.println("\n\n-------After Counting Frequency---------");

        final Map<String, Integer> frequencyMap = new HashMap<String, Integer>();

        for (int k = 0; k < removestopList.size(); k++) {

            String currentWord = removestopList.get(k);

            Integer frequency = frequencyMap.get(currentWord);

            // Add the word if it doesn't already exist, otherwise increment the frequency

            // counter.

            if (frequency == null) {

                frequency = 0;

            }

            frequencyMap.put(currentWord, frequency + 1);

        }

        Iterator<Map.Entry<String, Integer>> entries = frequencyMap.entrySet().iterator();

        while (entries.hasNext()) {

            Map.Entry<String, Integer> entry = entries.next();

            String key = entry.getKey();

            Integer value = entry.getValue();

            System.out.println(key + " = " + value);

        }

    }

    private static void suffixesString(ArrayList<String> removestopList) {

        System.out.println("\n\n--------After Removing Suffixes---------");

        for (int k = 0; k < removestopList.size(); k++) {

            String suffixString = removestopList.get(k);

            int flag = 0;

            for (int m = 0; m < suffixes.length; m++) {

                if (suffixString.endsWith(suffixes[m])) {

                    int len = suffixString.length();

                    int len1 = suffixes[m].length();

                    int len2 = len - len1;

                    String sufString = suffixString.substring(0, len2);

                    System.out.print(suffixString + "\t\t");

                    System.out.println(sufString);

                    flag = 1;

                    break;

                }

            }

            if (flag != 1)

                System.out.println(suffixString + "\t\t" + suffixString);

        }

    }

}

**Input.txt**

This is a sample input file. It contains some words and punctuation, like this word.

We are testing the conflation program. It should remove stopwords and count word frequencies properly.

**Output :**

Enter a filename:

LP-III\com\cl\conflation\Input.txt

--------------After Removing Stop Words-----------------

This

sample

input

file

contains

some

words

punctuation,

like

word

--------After Removing Suffixes---------

This This

sample sample

input input

file file

contains contains

some some

words words

punctuation, punctuation,

like like

word word

-------After Counting Frequency---------

input = 1

some = 1

contains = 1

file = 1

like = 1

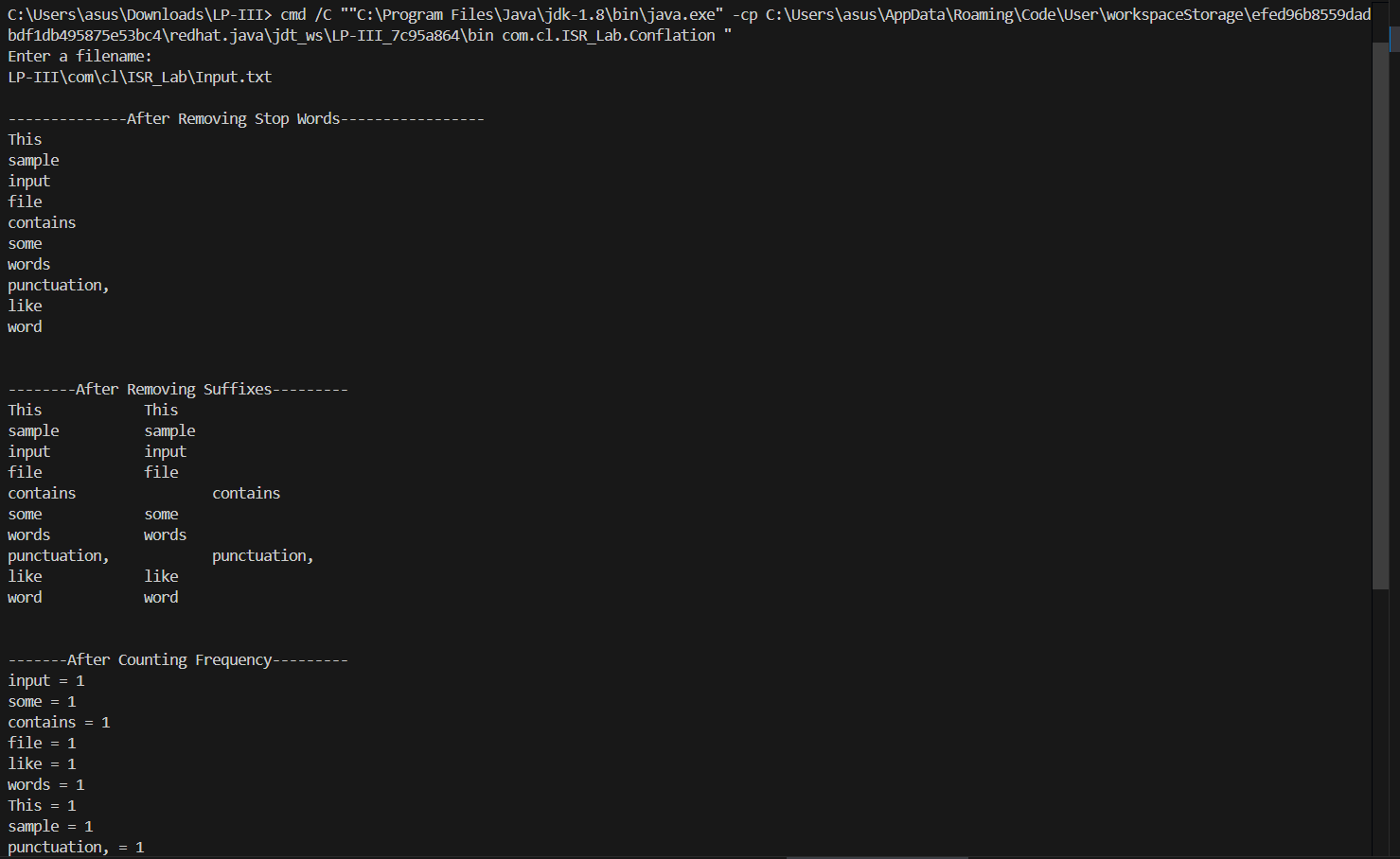
words = 1

This = 1

sample = 1

punctuation, = 1

word = 1



***Assignment-2 : Single Pass Clustering Algorithm***

**Input:**

package com.cl.ISR\_Lab;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.ArrayList;

public class SinglePass {

    public static void main(String[] args) throws IOException {

        BufferedReader stdInpt = new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Enter the number of Tokens:");

        int noOfDocuments = Integer.parseInt(stdInpt.readLine());

        System.out.println("Enter the number of Documents:");

        int noOfTokens = Integer.parseInt(stdInpt.readLine());

        System.out.println("Enter the threshold:");

        float threshold = Float.parseFloat(stdInpt.readLine());

        System.out.println("Enter the Document Token Matrix:");

        int[][] input = new int[noOfDocuments][noOfTokens];

        for (int i = 0; i < noOfDocuments; i++) {

            for (int j = 0; j < noOfTokens; j++) {

                System.out.println("Enter (" + i + "," + j + "):");

                input[i][j] = Integer.parseInt(stdInpt.readLine());

            }

        }

        SinglePassAlgorithm(noOfDocuments, noOfTokens, threshold, input);

    }

    private static void SinglePassAlgorithm(int noOfDocuments, int noOfTokens, float threshold, int[][] input) {

        int[][] cluster = new int[noOfDocuments][noOfDocuments + 1];

        ArrayList<Float[]> clusterRepresentative = new ArrayList<>();

        cluster[0][0] = 1;

        cluster[0][1] = 0;

        int noOfClusters = 1;

        Float[] temp = new Float[noOfTokens];

        temp = convertIntArrToFloatArr(input[0]);

        clusterRepresentative.add(temp);

        for (int i = 1; i < noOfDocuments; i++) {

            float max = -1;

            int clusterId = -1;

            for (int j = 0; j < clusterRepresentative.size(); j++) {

                float similarity = calculateSimilarity(convertIntArrToFloatArr(input[i]), clusterRepresentative.get(j));

                if (similarity > threshold) {

                    if (similarity > max) {

                        max = similarity;

                        clusterId = j;

                    }

                }

            }

            if (max == -1) {

                cluster[noOfClusters][0] = 1;

                cluster[noOfClusters][1] = i;

                noOfClusters++;

                clusterRepresentative.add(convertIntArrToFloatArr(input[i]));

            } else {

                cluster[clusterId][0] += 1;

                int index = cluster[clusterId][0];

                cluster[clusterId][index] = i;

                clusterRepresentative.set(clusterId, calculateClusterRepresentative(cluster[clusterId], input, noOfTokens));

            }

        }

        for (int i = 0; i < noOfClusters; i++) {

            System.out.print("\n" + i + "\t");

            for (int j = 1; j <= cluster[i][0]; j++) {

                System.out.print(" " + cluster[i][j]);

            }

        }

    }

    private static Float[] convertIntArrToFloatArr(int[] input) {

        int size = input.length;

        Float[] answer = new Float[size];

        for (int i = 0; i < size; i++) {

            answer[i] = (float) input[i];

        }

        return answer;

    }

    private static float calculateSimilarity(Float[] a, Float[] b) {

        float answer = 0;

        for (int i = 0; i < a.length; i++) {

            answer += a[i] \* b[i];

        }

        return answer;

    }

    private static Float[] calculateClusterRepresentative(int[] cluster, int[][] input, int noOfTokens) {

        Float[] answer = new Float[noOfTokens];

        for (int i = 0; i < noOfTokens; i++) {

            answer[i] = Float.parseFloat("0");

        }

        for (int i = 1; i <= cluster[0]; i++) {

            for (int j = 0; j < noOfTokens; j++) {

                answer[j] += input[cluster[i]][j];

            }

        }

        for (int i = 0; i < noOfTokens; i++) {

            answer[i] /= cluster[0];

        }

        return answer;

    }

}

**Ouput:**

Enter (3,0):

0

Enter (3,1):

3

Enter (3,2):

1

Enter (3,3):

0

Enter (3,4):

5

Enter (4,0):

1

Enter (4,1):

0

Enter (4,2):

1

Enter (4,3):

0

Enter (4,4):

1

0 0 1 3

1 2

2 4

Enter the number of Tokens:

5

Enter the number of Documents:

5

Enter the threshold:

10

Enter the Document Token Matrix:

Enter (0,0):

1

Enter (0,1):

3

Enter (0,2):

3

Enter (0,3):

2

Enter (0,4):

2

Enter (1,0):

2

Enter (1,1):

1

Enter (1,2):

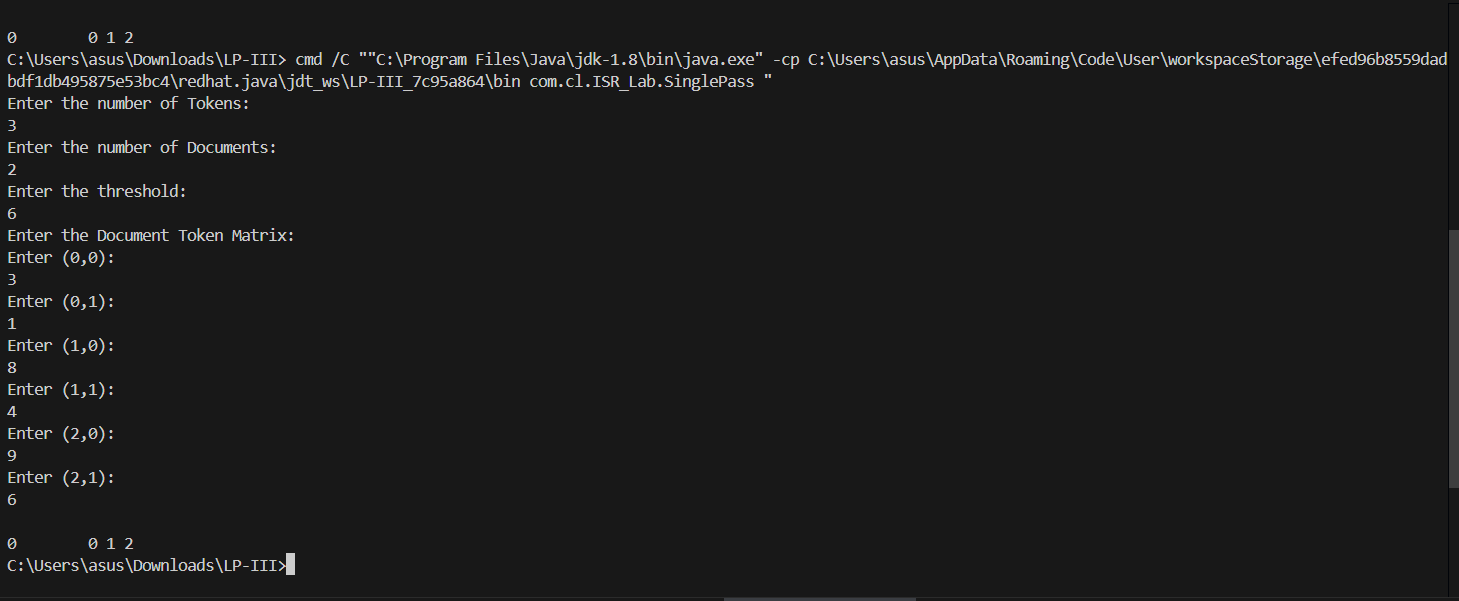
0

Enter (1,3):

1

Enter (1,4):

2



Enter (2,0):

0

Enter (2,1):

2

Enter (2,2):

0

Enter (2,3):

0

Enter (2,4):

1

***Assignment-3 : Retrieval of docs using Inverted files***

**Input:**

package com.cl.ISR\_Lab;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.ArrayList;

public class InvertedFile {

    public static void displayIndex(ArrayList<String> invertedData, int[][] docno) {

        int i, j;

        for (i = 0; i < invertedData.size(); i++) {

            System.out.print(invertedData.get(i) + "\t");

            for (j = 1; j <= docno[i][0]; j++) {

                System.out.print(docno[i][j] + "\t");

            }

            System.out.print("\n");

        }

    }

    public static void indexing(String fname, ArrayList<String> invertedData, int[][] docno, int fileno) {

        BufferedReader br;

        try {

            br = new BufferedReader(new FileReader(fname));

            String data = "", line = br.readLine();

            while (line != null) {

                data += line + " ";

                line = br.readLine();

            }

            String[] st = data.split("[ ,.]");

            String currenttoken = null;

            int i = 0;

            while (i < st.length) {

                currenttoken = st[i];

                int indx = invertedData.indexOf(currenttoken);

                if (indx == -1) {

                    invertedData.add(currenttoken);

                    indx = invertedData.indexOf(currenttoken);

                    docno[indx][0] = 1;

                    docno[indx][1] = fileno;

                } else {

                    docno[indx][docno[indx][0] + 1] = fileno;

                    docno[indx][0] += 1;

                }

                i += 1;

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

    public static void main(String[] args) throws NumberFormatException, IOException {

        String fname = "";

        ArrayList<String> invertedData = new ArrayList<>();

        int docno[][] = new int[100][10];

        InputStreamReader ins = new InputStreamReader(System.in);

        BufferedReader br = new BufferedReader(ins);

        System.out.println("\nENTER TOTAL NO OF FILES:");

        int no = Integer.parseInt(br.readLine());

        int i = 1;

        while (i <= no) {

            System.out.println("\nENTER FILE " + i + " NAME:");

            fname = br.readLine();

            indexing(fname, invertedData, docno, i);

            i += 1;

        }

        displayIndex(invertedData, docno);

    }

}

**InputFiles:**

**Anil.txt** how are you anil kumar

**Anil2.txt** how are you anil kumar and hi.

**Ouput:**

ENTER TOTAL NO OF FILES:

2

ENTER FILE 1 NAME:

LP-III\com\cl\ISR\_Lab\anil.txt

ENTER FILE 2 NAME:

LP-III\com\cl\ISR\_Lab\anil2.txt

how 1 2

are 1 2

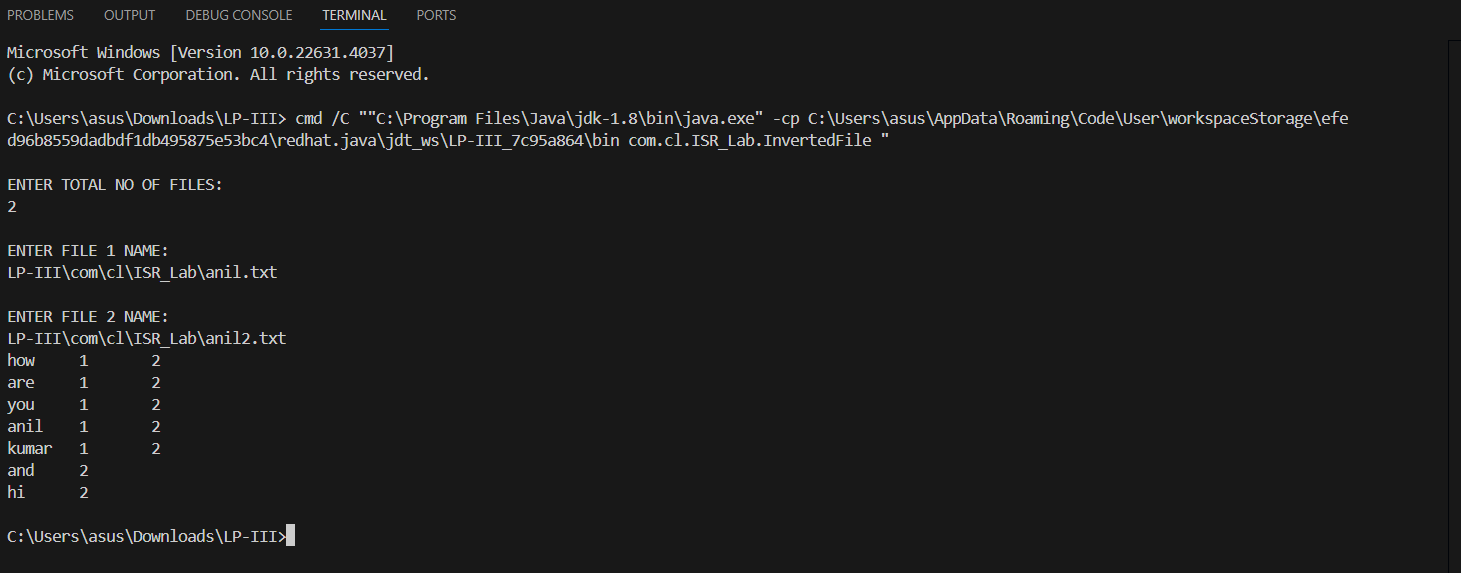
you 1 2

anil 1 2

kumar 1 2

and 2

hi 2



***Group B***

***Assignment-1 : Cal. Precision & Recall***

**Input:**

package com.cl.ISR\_Lab;

import java.util.HashSet;

import java.util.Set;

public class B1\_PrecisionRecallCalculator {

    // Method to calculate precision and recall

    public static void calculatePrecisionRecall(Set<String> retrievedDocs, Set<String> relevantDocs) {

        // Edge case: If no documents are retrieved, precision is undefined, set to 0

        if (retrievedDocs.isEmpty()) {

            System.out.println("Precision: 0.00");

            System.out.println("Recall: 0.00");

            //System.out.println("F1 Score: 0.00");

            return;

        }

        // Calculate true positives (intersection of retrieved and relevant documents)

        Set<String> truePositives = new HashSet<>(retrievedDocs);

        truePositives.retainAll(relevantDocs);

        // Precision: True Positives / Retrieved Documents

        double precision = (double) truePositives.size() / retrievedDocs.size();

        // Recall: True Positives / Relevant Documents

        double recall = relevantDocs.isEmpty() ? 0 : (double) truePositives.size() / relevantDocs.size();

        // F1 Score: 2 \* (Precision \* Recall) / (Precision + Recall)

        double f1Score = (precision + recall == 0) ? 0 : 2 \* (precision \* recall) / (precision + recall);

        // Print the results

        System.out.printf("Precision: %.2f%n", precision);

        System.out.printf("Recall: %.2f%n", recall);

        //System.out.printf("F1 Score: %.2f%n", f1Score);

    }

    public static void main(String[] args) {

        // Sample input for query q1

        Set<String> retrievedDocs = new HashSet<>();

        retrievedDocs.add("doc1");

        retrievedDocs.add("doc2");

        retrievedDocs.add("doc3");

        retrievedDocs.add("doc4");

        Set<String> relevantDocs = new HashSet<>();

        relevantDocs.add("doc2");

        relevantDocs.add("doc3");

        relevantDocs.add("doc5");

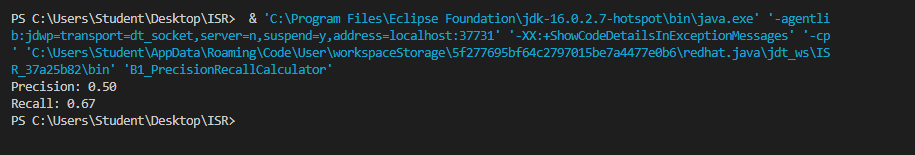
        // Call the function to calculate precision and recall

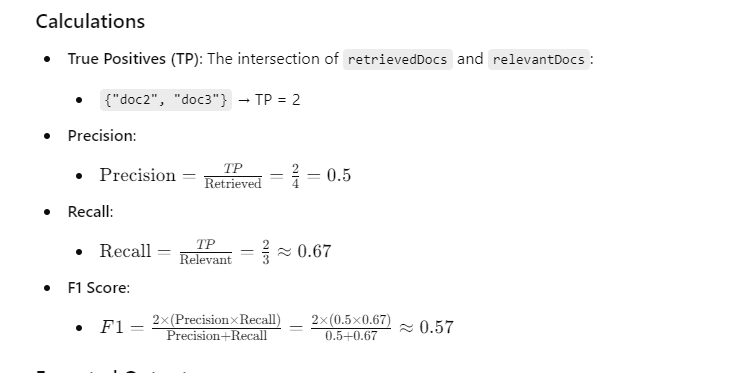
        calculatePrecisionRecall(retrievedDocs, relevantDocs);

    }

}

**Output:**





***Assignment-2 :***

***Cal. Harmonic Mean(F-measure) & E-Measure***

**Input:**

package com.cl.ISR\_Lab;

import java.util.HashSet;

import java.util.Set;

public class b2\_PrecisionRecallCalculator\_HM {

    // Method to calculate precision and recall

    public static void calculatePrecisionRecall(Set<String> retrievedDocs, Set<String> relevantDocs) {

        // Edge case: If no documents are retrieved, precision and recall are 0

        if (retrievedDocs.isEmpty()) {

            System.out.println("Precision: 0.00");

            System.out.println("Recall: 0.00");

            System.out.println("F1-measure: 0.00");

            System.out.println("E-measure: 0.00");

            return;

        }

        // Calculate true positives (intersection of retrieved and relevant documents)

        Set<String> truePositives = new HashSet<>(retrievedDocs);

        truePositives.retainAll(relevantDocs);

        // Precision: True Positives / Retrieved Documents

        double precision = (double) truePositives.size() / retrievedDocs.size();

        // Recall: True Positives / Relevant Documents

        double recall = relevantDocs.isEmpty() ? 0 : (double) truePositives.size() / relevantDocs.size();

        // F1-Measure: Harmonic mean of Precision and Recall

        double f1 = (precision + recall > 0) ? 2 \* ((precision \* recall) / (precision + recall)) : 0;

        // E-measure: Effectiveness measure with alpha = 0.5 (equal weight to precision and recall)

        double alpha = 0.5;

        double eMeasure = (precision > 0 && recall > 0)

                          ? 1 - (1 / ((alpha / precision) + ((1 - alpha) / recall)))

                          : 0;

        // Print the results

        System.out.printf("Precision: %.2f%n", precision);

        System.out.printf("Recall: %.2f%n", recall);

        System.out.printf("F1-Measure: %.2f%n", f1);

        System.out.printf("E-Measure: %.2f%n", eMeasure);

    }

    public static void main(String[] args) {

        // Sample input for query q1

        Set<String> retrievedDocs = new HashSet<>();

        retrievedDocs.add("doc1");

        retrievedDocs.add("doc2");

        retrievedDocs.add("doc3");

        retrievedDocs.add("doc4");

        Set<String> relevantDocs = new HashSet<>();

        relevantDocs.add("doc2");

        relevantDocs.add("doc3");

        relevantDocs.add("doc5");

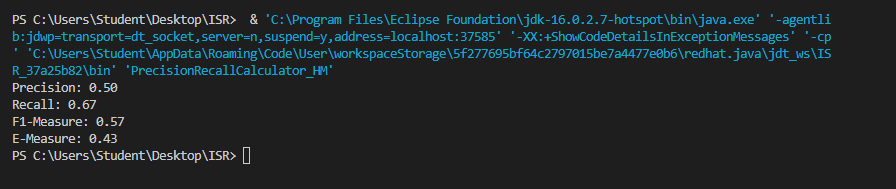
        // Call the function to calculate precision, recall, F1-measure, and E-measure

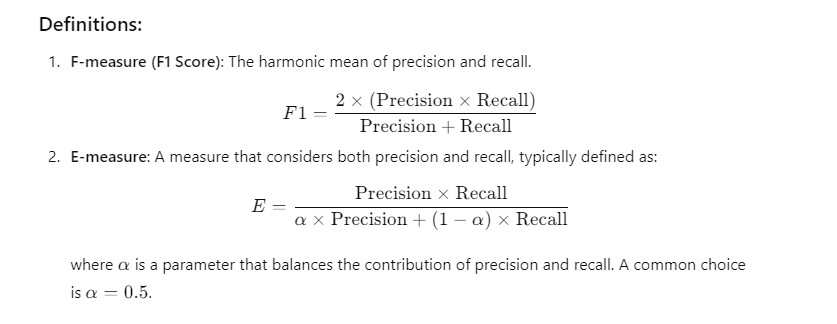
        calculatePrecisionRecall(retrievedDocs, relevantDocs);

    }

}

**Output:**

****



***Assignment-3 : Feature Extraction in 2D Color Image***

**Input:**

import matplotlib.pyplot as plt # Importing matplotlib for image display and plotting

import cv2 # Importing OpenCV for image processing

from skimage.color import rgb2gray # Importing function to convert image to grayscale

from skimage.filters import threshold\_otsu, gaussian # Importing Otsu thresholding and Gaussian filter

from skimage.io import imread # Importing function to read images

# Load and display the original image

image = imread(r"C:\\Users\\asus\\Downloads\\suhail\\ISR LAb\\123.jpg")

plt.imshow(image)

plt.title("Original Image")

plt.axis('off')

plt.show()

# Create red and yellow intensity images

red, yellow = image.copy(), image.copy()

# Set green and blue channels to 0 for red image

red[:, :, (1, 2)] = 0

# Set blue channel to 0 for yellow image (leaving only red and green)

yellow[:, :, 2] = 0

# Display red and yellow intensity images

fig, axes = plt.subplots(1, 2, figsize=(10, 5))

axes[0].imshow(red)

axes[0].set\_title("Red Intensity")

axes[0].axis('off')

axes[1].imshow(yellow)

axes[1].set\_title("Yellow Intensity")

axes[1].axis('off')

plt.show()

# Convert the image to grayscale

gray\_image = rgb2gray(image)

# Display the original and grayscale images side by side

fig, axes = plt.subplots(1, 2, figsize=(10, 5))

axes[0].imshow(image)

axes[0].set\_title("Color Image")

axes[0].axis('off')

axes[1].imshow(gray\_image, cmap='gray')

axes[1].set\_title("Grayscale Image")

axes[1].axis('off')

plt.show()

# Print the dimensions of the images

print("Colored image shape:", image.shape)

print("Grayscale image shape:", gray\_image.shape)

# Apply Otsu's thresholding to the grayscale image

thresh = threshold\_otsu(gray\_image)

binary\_image = gray\_image > thresh

# Display grayscale and binary (thresholded) images

fig, axes = plt.subplots(1, 2, figsize=(10, 5))

axes[0].imshow(gray\_image, cmap='gray')

axes[0].set\_title("Grayscale Image")

axes[0].axis('off')

axes[1].imshow(binary\_image, cmap='gray')

axes[1].set\_title("Otsu Binary Image")

axes[1].axis('off')

plt.show()

# Apply Gaussian blur to the grayscale image

blurred\_image = gaussian(gray\_image, sigma=20)

# Display the grayscale image and the blurred image

fig, axes = plt.subplots(1, 2, figsize=(10, 5))

axes[0].imshow(gray\_image, cmap='gray')

axes[0].set\_title("Grayscale Image")

axes[0].axis('off')

axes[1].imshow(blurred\_image, cmap='gray')

axes[1].set\_title("20 Sigma Blurred Image")

axes[1].axis('off')

plt.show()

# Example: Reading and plotting histogram of an image using OpenCV and Matplotlib

img = cv2.imread('ex.jpg', 0) # Load the image in grayscale

# Calculate the histogram for grayscale image (0-255 intensity levels)

histg = cv2.calcHist([img], [0], None, [256], [0, 256])

# Plot the histogram to analyze pixel intensity distribution

plt.plot(histg)

plt.title('Histogram of Grayscale Image')

plt.xlabel('Pixel Intensity')

plt.ylabel('Frequency')

plt.show()

import cv2

import numpy as np

import matplotlib.pyplot as plt

from skimage import io

image\_path = "123.jpg" # Replace with your image path

image = cv2.imread(image\_path)

image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

def plot\_color\_histogram(image):

color = ('r', 'g', 'b')

plt.figure(figsize=(12, 6))

for i, col in enumerate(color):

hist = cv2.calcHist([image], [i], None, [256], [0, 256])

plt.plot(hist, color=col)

plt.xlim([0, 256])

plt.title('Color Histogram')

plt.xlabel('Pixel Intensity')

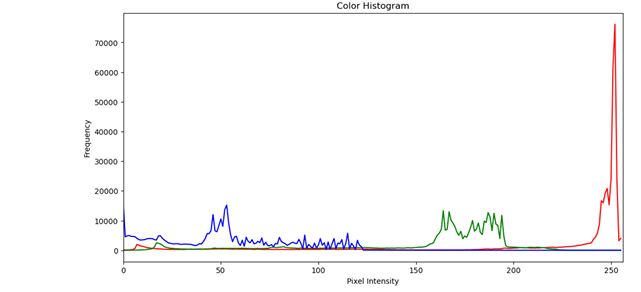
plt.ylabel('Frequency')

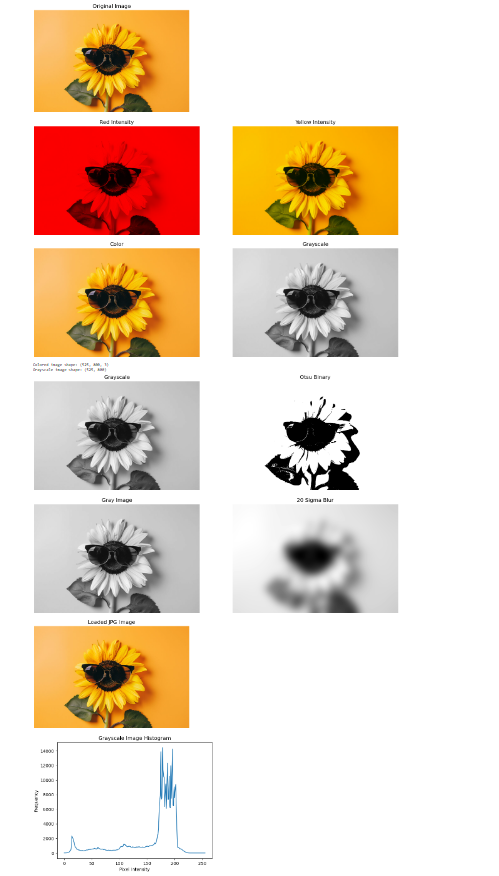
plt.show()

# Plot color histogram

plot\_color\_histogram(image\_rgb)

**Output:**





***Group C***

***Assignment-1 : Web Crawler***

**Input:**

import requests

from bs4 import BeautifulSoup

headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36'}

request = requests.get("https://www.amazon.in/s?k=mobile+phone+under+20000&crid=2J6DM5J2AVMGE&sprefix=mobile%2Caps%2C256&ref=nb\_sb\_ss\_ts-doa-p\_3\_63",headers=headers)

# print(request)

# print(request.content)

soup = BeautifulSoup(request.content, 'html.parser')

# print(soup.prettify())

products = []

product\_elements = soup.select('.s-main-slot .s-result-item')

for product in product\_elements:

title = product.select\_one('h2 a span').get\_text(strip=True) if product.select\_one('h2 a span') else 'no Title'

price = product.select\_one('.a-price').get\_text(strip=True).strip() if product.select\_one('.a-price') else 'No price'

products.append({

'title': title,

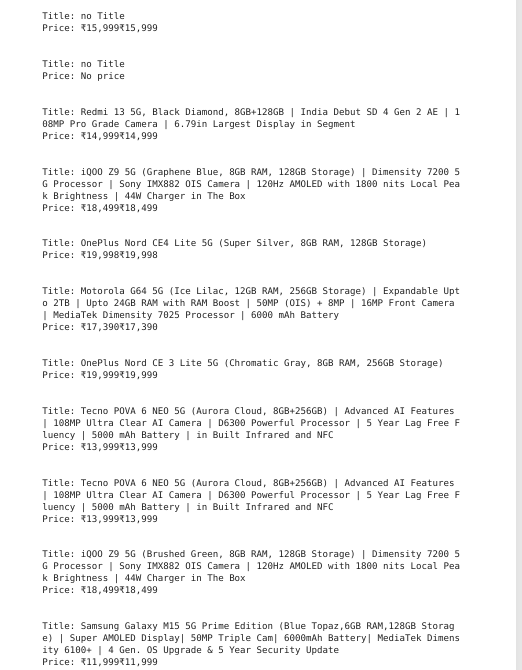
'price': price

})

for product in products:

print(f"Title: {product['title']} \nPrice: {product['price']}\n\n")

**Output:**



***Assignment-2 : Weather Report***

**Input:**

import requests

import json

def get\_weather\_report():

# API URL

API\_URL = "https://api.weatherapi.com/v1/current.json?key=0ffbc5c35b604366adb42044240210&q="

# Get the city name from the user

city\_name = input("Enter City Name To Get Weather Report: ")

# Append the city name to the API URL

full\_api\_url = API\_URL + city\_name

try:

# Send the GET request

response = requests.get(full\_api\_url)

# Check if the request was successful (status code 200)

if response.status\_code == 200:

# Parse the JSON response

json\_response = response.json()

# Extract necessary fields from JSON response

temperature = json\_response['current']['temp\_c']

wind\_speed = json\_response['current']['wind\_kph']

description = json\_response['current']['condition']['text']

city = json\_response['location']['name']

state = json\_response['location']['region']

country = json\_response['location']['country']

# Print the formatted weather report

print(f"Weather in ({city}, {state}, {country}):")

print(f"Temperature: {temperature}°C")

print(f"Wind Speed: {wind\_speed} kph")

print(f"Condition: {description}")

else:

print(f"Something went wrong... HTTP Status Code: {response.status\_code}")

except requests.exceptions.RequestException as e:

# Handle connection errors or other exceptions

print(f"An error occurred: {e}")

# Call the function to get weather report

get\_weather\_report()

**Output:**

