Text Summarisation Tool

*Declaring Variables:*

static Dictionary<string, int> countOfWords = new Dictionary<string, int>();

static List<String> sentences = new List<string>();

static List<List<String>> wordListInSentenceLists = new List<List<string>>();

static List<string> stopwords = new List<string>();

static string sumFile = "Test.txt";

static string stopFile = "stopwords.txt";

//Defualt summarisation factor = 50%

static float sumFact = 50f;

static List<List<string>> sentencesOut = new List<List<string>>();

static int lengthOfSumWords = 0;

static float sf = 0f; //for calculating summarised factor

// Name of file writing out to

static string outputFileName = "summary.txt";

//If stopword list contains current string

private static bool containsStopwords(string str)

{

return stopwords.Contains(str.ToLower());

}

//Temporary Variables

static string sumFileTemp = "";

static string stopFileTemp = "";

static float sumFactTemp = 0f;

*Main()*

GetUserInput();

// Characters that split sentences to words.

char[] splitForWords = " ,./<>?;\':\"[]{}`~!@#$%^&\*()\_+|\\=-1234567890".ToArray();

// Characters that splits whole text to sentences.

string[] splitForSentences = { ". " };

// Reading all files.

string tempString = "";

StreamReader sr = null;

// Reading StopWords.txt and remove end markers.

sr = new StreamReader(new FileStream(stopFileTemp, FileMode.Open));

tempString = sr.ReadToEnd().Replace("\n", " ").Replace("\r", " ");

// Removing duplicated strings making each word in list unique

stopwords.AddRange(tempString.Split(splitForWords).Distinct());

stopwords = stopwords.Select(x => x.ToLower()).ToList();

// Read input file, remove end markers and process file, removing any empty strings.

sr = new StreamReader(new FileStream(sumFileTemp, FileMode.Open));

tempString = sr.ReadToEnd().Replace("\n", " ").Replace("\r", " ");

sentences.AddRange(tempString.Split(splitForSentences, StringSplitOptions.RemoveEmptyEntries));

// Loop each sentence and split into words

foreach (string sentence in sentences)

{

wordListInSentenceLists.Add(sentence.Split(splitForWords, StringSplitOptions.RemoveEmptyEntries).Select(x => x.ToLower()).ToList());

}

//Create dictionary to hold word and it's occurences

List<string> wordsList = new List<string>();

// Splitting words by appropriate characters

wordsList.AddRange(tempString.Split(splitForWords, StringSplitOptions.RemoveEmptyEntries));

// Remove stopwords, convert to lower case & add to dictionary

wordsList.RemoveAll(containsStopwords);

countOfWords = wordsList.GroupBy(x => x.ToLower()).ToDictionary(g => g.Key, g => g.Count());

//Sort Dict

countOfWords = countOfWords.OrderByDescending(pair => pair.Value).ToDictionary(x => x.Key, x => x.Value);

//Index of summarised sentences in original sentence list

Console.WriteLine("Count-----> Word");

Console.ReadLine();

// Write keywords and their occurences

foreach (KeyValuePair<string, int> pair in countOfWords)

{

Console.WriteLine("[{0}] : {1}", pair.Value, pair.Key);

}

Process();

Output();

*GetUserInput()*

while (true)

{

/\* Get User file \*/

// Get user to input file name

Console.WriteLine("Please enter the name of the file you would like processing: ");

string userInput = Console.ReadLine();

// if user doesn't enter a value..... get default

if (userInput != "")

sumFileTemp = userInput;

else

{

sumFileTemp = sumFile;

}

// if used input == Q or q, exit program.

if (userInput == "Q" || userInput == "q") return;

/\* Get Stopwords \*/

stopFileTemp = stopFile;

/\* Get Sumarisation Factor \*/

Console.Write("Summarisation Factor: ");

string userSumFactor = Console.ReadLine();

// if user doesn't enter a value..... get default

if (userSumFactor == "")

{

sumFactTemp = sumFact;

}

else

{

sumFactTemp = float.Parse(userSumFactor);

}

if (userSumFactor == "Q" || userSumFactor == "q") return;

// if files exist quit while loop

// if it doesn't then re-loop

if (File.Exists(sumFileTemp) && File.Exists(stopFileTemp))

{

break;

}

else

{

Console.Write("No file exist: " + sumFileTemp + " or " + stopFileTemp);

}

}

// Write out file names to user

Console.WriteLine("infile name : " + sumFileTemp);

Console.WriteLine("stopwords filename : " + stopFileTemp);

Console.ReadLine();

*Process()*

// Get word occurence for each word in the dictionary

foreach (KeyValuePair<string, int> occurredWord in countOfWords)

{

List<int> indexlist = new List<int>();

int maxcount = 0;

int index = 0;

//Size of word list

int userWordLength = wordListInSentenceLists.Count;

string maxSentence = "";

int maxIndex = 0;

// Finding sentence with most occurences of key words

foreach (List<string> wordlist in wordListInSentenceLists)

{

// Get occurrence of word

int count = wordlist.Count(x => x == occurredWord.Key);

// If word has max occurence save word, count and index

if (count > maxcount)

{

maxSentence = sentences[index];

maxIndex = index;

maxcount = count;

}

index++;

}

// If most occured found..

if (maxcount > 0)

{

lengthOfSumWords++;

// Result list

List<string> resultStringList = new List<string>();

// Get actual index of found sentence as sentence will be changed

int actualIndex = maxIndex;

foreach (int identified in indexlist)

{

if (actualIndex >= identified) //index

actualIndex++;

}

indexlist.Add(actualIndex);

indexlist.Sort();

// Add to output

resultStringList.AddRange(new string[] { maxcount.ToString(), occurredWord.Key, maxSentence });

sentencesOut.Add(resultStringList);

// Take out summarised sentence.

wordListInSentenceLists.RemoveAt(maxIndex); //list of wordlist

sentences.RemoveAt(maxIndex);//list of sentence

}

// Calculate summarisation factor. If it's greater than the user requested SF, break

sf = 100f \* lengthOfSumWords / userWordLength;

if (sf >= sumFactTemp)

{

break;

}

}

*Output()*

Console.WriteLine("Summarised file reads: ");

Console.ReadLine();

List<string> summarisedDoc = new List<string>();

foreach (List<string> ans in sentencesOut)

{

// Output each element

summarisedDoc.Add(string.Join(",", ans));

Console.Write(ans[0] + " : " + ans[1].PadRight(13) + " : " + ans[2].Substring(0, Math.Min(50, ans[2].Length)));

}

// Write real summarisation factor

Console.ReadLine();

Console.WriteLine("Actual SF = " + sf.ToString("0.00"));

// Save to file

File.WriteAllLines(outputFileName, summarisedDoc);

Console.ReadLine();

Console.WriteLine("Saved to file : " + outputFileName);

Console.ReadKey();